PFC – Auditoría de aplicaciones web: metodología y práctica profesional.

Memoria

02/01/2015

Universidad:

Universitat Oberta de Catalunya — Estudios de segundo ciclo de Ingeniería en Informática.

Área del proyecto: Seguridad Informática.

Curso: 2014 – 2015 Primer semestre.

Estudiante:

Miriam Rodríguez Sánchez

Tutor:

Cristina Pérez Solà

RESUMEN

La finalidad del presente trabajo es consolidar los conocimientos adquiridos durante la carrera, y más en concreto, en el área de seguridad y administración de redes y sistemas operativos.

Esta memoria es el documento que recoge todo el trabajo realizado durante el proyecto, en el que se refleja tanto el estudio teórico del tema tratado, como la realización práctica del ejercicio de auditoría propuesto.

En el proyecto, se plantea la problemática de una pequeña empresa que busca mejorar el desarrollo de su software, para diseñar páginas webs más seguras. La inseguridad de las páginas webs es un problema actual del mundo real. Poner solución a esto requiere de una labor técnica, que se condensa en una auditoría de la aplicación. El proceso de auditoría web que se describe aquí, nos servirá para detectar y resolver las vulnerabilidades del código, o de la configuración del sistema que aloja la aplicación.

ABSTRACT

The purpose of this paper is to consolidate the knowledge acquired throughout the academic career, and more specifically in the area of security, network management and operating systems.

This project report is the document containing all the work done during the project, in which both the theoretical study of the topic, as the practical realization of the proposed audit exercise is reflected.

In the present project, the goal of a software company is to improve their software development to design safer web applications. The insecurity of the websites is an ongoing problem in the real world. Solving this problem requires a technical work, which is condensed in an audit of the application. The web audit process described here could be used to detect and solve vulnerabilities in the application code or within the configuration of the system hosting the application.

TABLA DE CONTENIDOS

TABLA DE ILUSTRACIONES

COLABORACIONES Y AGRADECIMIENTOS

Doy las gracias a las personas que han aceptado colaborar para ayudarme a realizar este proyecto: a Rafael Alfaro March por su aportación de conocimientos prácticos basados en su experiencia laboral y su labor como consultor de empresa, y a Alejandro Murillo Alapont por asumir el rol de cliente y darme permiso para ejecutar los ejercicios prácticos de la auditoría sobre varias webs alojadas en uno de sus servidores, pudiendo plasmar así en el proyecto la resolución de un caso real.

CAPÍTULO 1. INTRODUCCIÓN

Desde el primer instante en que una aplicación web es publicada en la red, es susceptible de recibir ataques malintencionados. Prevenir que esos ataques resulten exitosos será la principal función de una auditoría web.

Una auditoría web (o auditoría informática [1]), es un proceso de revisión de la seguridad de una aplicación web, cuya finalidad es encontrar las vulnerabilidades de seguridad existentes, con el objetivo de solucionarlas antes de que el atacante se aproveche de las mismas.

Un auditor es una persona con conocimientos técnicos profundos en el ámbito de la informática, que es capaz de analizar una aplicación desde el punto de vista de un atacante anónimo.

En este proyecto vamos a aplicar los conocimientos aprendidos durante la carrera, en el área de redes y seguridad, para resolver un caso práctico real de auditoría web desde una perspectiva profesional.

Partimos de la solicitud de una pequeña empresa, dedicada a la creación de páginas web, para revisar la seguridad de uno de sus servidores, en el que aloja tres de sus aplicaciones web, a través de una auditoría web externa. Resolveremos la auditoría solicitada tomando como base las metodologías abiertas existentes más conocidas, como OWASP y OSSTMM que explicaremos más adelante, cuyo uso está muy extendido entre la comunidad técnica dedicada a la seguridad.

1.1. CONTEXTO EN QUE SE DESARROLLA

Hasta hace pocos años no era posible plantearse la necesidad de realizar**¡Error! Marcador no definido.** auditorías sobre las aplicaciones informáticas con el fin de evitar ataques externos por red. La razón es simple: porque hasta los años 80 no se desarrolló internet tal como lo conocemos hoy día [2].

En aquellos años, el uso de ordenadores personales apenas estaba extendido y los pocos que había, (fuera de ciertos ámbitos como el de defensa, o el universitario), no estaban conectados a ninguna red. Sus funciones y capacidades eran muy limitadas. La única forma de acceder ilegítimamente a un ordenador personal era, o bien ganar acceso físico a la habitación donde se encontrara la máquina, o bien, que su dueño insertara en él un dispositivo externo como un disquete, previamente infectado. Así pues, el papel de la seguridad de las aplicaciones no era tan importante como lo es hoy.

A partir de los años 80 se crea el estándar Ethernet, y se formalizan protocolos como TCP/IP, o DNS. Hasta los 90 no aparece el WWW. Es en ésta década cuando se abre la red al comercio y se dispara el número de ordenadores conectados.

Con el avance tan veloz de las tecnologías TIC, en pocos años se pasa de conexiones no permanentes, a través de llamadas por líneas telefónicas, a conexiones por cable o fibra óptica permanentes, que no inhabilitan la línea de teléfono. Las tarifas se reducen para aumentar la demanda, y se pasa de pagar por tiempo de conexión, o cantidad de megas descargados, a una tarifa plana mensual, que hace internet más asequible para todo el mundo. Las páginas web

evolucionan rápidamente y pasan de ser sitios estáticos, que ofrecen texto informativo y poco más, a sitios con contenido dinámico que ejecutan aplicaciones y devuelven sus resultados.

Así llegamos al momento actual, en el que la mayoría de las empresas, grandes y pequeñas, ya no sólo tienen presencia en internet, sino que la red se ha convertido en otro escaparate en el que ofrecer sus servicios; Internet ha llegado a ser un nuevo medio de comunicación, y todas las televisiones, periódicos y radios tienen su canal o web donde retransmiten sus contenidos. Además, la tipología de red ha saltado del cableado clásico, a la Wifi inalámbrica, y los desarrolladores de software están creando sistemas operativos y aplicaciones pensados para otros dispositivos de menor tamaño, distintos a los ordenadores de sobremesa, como los "tablets", los "ebooks", o los móviles. Gracias a esto, el uso de la red se ha extendido entre todo tipo de usuarios, y ya no se limita a jóvenes aficionados a la informática. Hoy día, todo el mundo lleva internet en su móvil, accede a aplicaciones susceptibles de ser atacadas, y por tanto, mantener una seguridad elevada es vital incluso para proteger la privacidad de las personas. Una de las piezas clave para certificar un buen nivel de seguridad, es la auditoria de las aplicaciones.

1.2. JUSTIFICACIÓN

DISPONIBILIDAD Y PRESENCIA ONLINE

Cada vez son más los negocios que operan por la red. Internet se ha convertido en un espacio donde las empresas ofrecen servicios de todo tipo, desde compras, recargas de móviles, almacén de datos, operaciones bancarias, reservas de billetes, viajes, o entradas para espectáculos, etc.

Todos estos servicios se dan habitualmente por medio de aplicaciones específicas, cuya interfaz de salida son las páginas web. De este modo, las páginas web se convierten en el nexo entre el usuario o cliente, y la empresa que ofrece un servicio.

También es habitual que muchas empresas decidan usar la red para darse a conocer y llegar a más cantidad de público, aunque no ofrezcan sus servicios "on line"; tener presencia en internet es igual de importante para su crecimiento, que dar servicios a través de la red, y no sólo para empresas: los artistas o personajes públicos también pueden encontrar en la red un modo de comunicarse con sus seguidores y anunciar sus trabajos. Esta presencia, en la mayoría de ocasiones, también se consigue por medio de una página web propia, (aparte de los perfiles en las redes sociales), que da a conocer al cliente potencial el servicio que la entidad ofrece.

Por estos motivos, es importante la auditoria, ya que la web se convierte en el primer punto de ataque a la entidad o al usuario. Es similar a la puerta de entrada a un local. Hacer de la web un sitio fiable para los usuarios que accedan es importante para la continuidad de la actividad. Una web insegura, y poco confiable, no sólo permite que tengan éxito ataques o robos a la empresa, sino también, en ocasiones, a los clientes que tengan contratado un servicio. Si un cliente sufre un revés a su confianza, a causa de operar en el espacio web de una empresa, con toda seguridad no volverá a utilizar este servicio, y no se lo aconsejará a nadie.

1.3. OBJETIVOS

Dividiremos los objetivos a alcanzar durante el trabajo final de carrera entre objetivos generales, y objetivos específicos, tal como sigue:

1.3.1. OBJETIVOS GENERALES DEL PROYECTO

Los tres objetivos generales son los siguientes:

- Hacer una síntesis de todos los conocimientos adquiridos tanto en la carrera como en el entorno de trabajo.
- Hacer una aproximación práctica al ejercicio profesional de la auditoría, principalmente web, pero en general también de aplicaciones móviles, de redes Wifi o infraestructuras. Para ello se llevará a cabo una auditoría a varias páginas web, propiedad de una empresa colaboradora a la que previamente hemos solicitado permiso y con la que hemos acordado la realización del ejercicio.
- Aprender como auditar aplicaciones web siguiendo las metodologías teóricas existentes. Conocer a fondo las metodologías mediante la aplicación práctica de las mismas en un caso de ejemplo real.

1.3.2. OBJETIVOS ESPECÍFICOS DEL PROYECTO

Los cuatro objetivos específicos son los siguientes:

- Partiendo de las metodologías existentes se intentará mejorar el proceso de auditoría, mediante la ayuda de profesionales que nos aporten un punto de vista práctico fruto de su propia experiencia laboral. El objetivo será obtener un proceso más eficiente, más práctico, y más dirigido a la práctica profesional real de la auditoría.
- Concienciar a las empresas de desarrollo de software sobre las ventajas de crear software seguro desde el inicio, que no sea vulnerable a los ataques de la red. Concienciar a los profesionales relacionados (desarrolladores y diseñadores de software) sobre la mayor efectividad del software seguro.
- Realizar la auditoría con herramientas de software libre, que se puedan encontrar disponibles en Internet.
- Ofrecer una guía de aprendizaje a estudiantes de la Ingeniería en Informática interesados en el campo de la auditoría de la seguridad o el desarrollo de software seguro. Para ello, se informará tanto de vulnerabilidades posibles en aplicaciones web, como de las recomendaciones para mitigarlas.

1.4. PRODUCTOS ENTREGABLES

Los productos a entregar previstos al finalizar el presente trabajo, son los siguientes:

- Informe final predefinido de resultados de la auditoria, en formato PDF. El informe incluirá las partes fundamentales de un proyecto de auditoría, y recogerá los resultados de las pruebas realizadas sobre la, o las, webs, un resumen del caso, las vulnerabilidades encontradas y las evidencias tomadas acerca de éstas, y además las recomendaciones técnicas para resolverlas[1].

[1] Como hay que respetar la confidencialidad del propietario de las aplicaciones, todas las evidencias y datos sensibles que contenga el informe se presentarán ofuscados, de modo que su divulgación no sea posible. Así, los datos confidenciales no podrán leerse, pero sí se podrán ver las vulnerabilidades que presente la aplicación.

- La memoria final en formato PDF, que muestra y explica todo el trabajo llevado a cabo durante el proyecto y cuya finalidad es cumplir los objetivos propuestos en el plan de trabajo inicial.

- La presentación en dos formatos: un PowerPoint, y un video explicativo. Ambos resumen todo el trabajo realizado en la memoria, y muestran los resultados obtenidos.

1.5. RECURSOS NECESARIOS PRELIMINARES

1.5.1. INFRAESTRUCTURA: HARDWARE Y SOFTWARE

1. **Hardware**.

   Se utilizarán dos equipos, aunque sólo uno de ellos es estrictamente necesario:

   a) Equipo de sobremesa con las siguientes características: procesador Intel Core i5 a 2.6GHz, 4Gb de memoria RAM y sistema operativo de 64 bits Windows 7 Ultimate SP1.
   b) Equipo portátil Lenovo 3000 N500, con las siguientes características: procesador Intel Core 2 Duo T7250 a 2GHz, 4Gb de memoria RAM, 250Gb de disco duro, y sistema operativo Windows XP Profesional SP3.

2. **Software**.

Se utilizará el sistema operativo Windows 7 como sistema base para hacer las pruebas de campo. Para poder utilizar herramientas de otros sistemas operativos, se utilizarán máquinas virtuales instaladas sobre VMware Player o Virtualbox. La máquina virtual que se usará será un Kali Linux (Ubuntu). Se utilizará el sistema operativo Windows XP para desarrollar la documentación en la que se recogerá la toma de evidencias y la memoria final.

1.5.2. HERRAMIENTAS

A pesar de que las herramientas de pago suelen ser más completas, ofrecen más facilidades, y presentan menos limitaciones, en el presente trabajo se emplearán exclusivamente herramientas de software libre, accesibles a través de la red, tanto para sistemas Windows como para sistemas Linux. Con esto comprobaríamos que cualquier persona malintencionada con conocimientos técnicos suficientes, que se proponga realizar ataques a sistemas, tiene la posibilidad de hacerlo sin contar con el apoyo de una entidad, sin adquirir costosas licencias de software, y sin pagar por herramientas especiales y de uso restringido.

1.5.2.1. HERRAMIENTAS PARA SISTEMAS WINDOWS

Se usarán principalmente las siguientes:

- **Navegadores** web de uso muy extendido: Internet Explorer y Mozilla Firefox.
- **Complementos** (plugins, addons) varios para Firefox, como foxyproxy.
- **Ssldigger**: para escanear los algoritmos SSL/TLS soportados por el servidor.
- **OWASP ZAP**: proxy para interceptar peticiones.
- **Microsoft office**: Word, y PowerPoint, para generar la memoria y la presentación.

1.5.2.2. HERRAMIENTAS PARA SISTEMAS LINUX

Se usarán principalmente las siguientes:

- **TLSSLED**: para escanear los algoritmos SSL/TLS soportados por el servidor.
- **Dirb**: para enumerar directorios en aplicaciones web.
- **Nikto**: para escanear vulnerabilidades en la aplicación.
- **Diccionarios de fuzzdb**: diccionarios para hacer fuzzing de las aplicaciones web.
- **SQLmap**: para detectar y explotar las inyecciones SQL.

1.6. DESCRIPCIÓN BREVE DE LA METODOLOGÍA

Para la elaboración de este proyecto, se va a utilizar una metodología basada en OWASP, OSSTMM, y una serie de buenas prácticas y consejos profesionales.

La fundación OWASP[2] es una organización sin ánimo de lucro fundada en EE.UU. Uno de sus proyectos más conocidos es el "OWASP Testing Project" el cual describe toda una metodología de pruebas para verificar la seguridad de las aplicaciones web. En general se suele utilizar el término metodología OWASP para referirse al testeo de aplicaciones web, pero OWASP abarca muchos más proyectos. En este proyecto utilizaremos la metodología definida en el proyecto de pruebas de OWASP.

El manual OSSTMM[3] (Manual de la Metodología Abierta de Testeo de Seguridad) describe el modo de realizar pruebas de seguridad, así como las métricas a emplear por profesionales al realizar las Auditorías de Seguridad. La fundación ISECOM (Institute for Security and Open Methodologies), sin ánimo de lucro, es la que publica y actualiza el manual. También ofrece formación y distintas certificaciones.

OSSTMM se ha convertido en un estándar de facto, ya que fue el primero en surgir. El manual describe los casos de prueba de los elementos que deben ser probados y como medir los resultados. Es más general que la metodología OWASP, porque describe todo tipo de auditoría, no sólo la de aplicaciones web, pero lo tomaremos como referencia para este proyecto.

1.6.1. ESTRUCTURA DEL PROYECTO: FASES PRINCIPALES

La metodología que seguiremos en nuestro caso se va a estructurar en cuatro fases: reconocimiento, mapeado, descubrimiento, y explotación.

No todas las fases se desarrollarán en orden estrictamente secuencial. Algunas están tan interrelacionadas que suelen desarrollarse en paralelo o solaparse en un solo paso. A continuación, se describen las cuatro fases:

1.6.1.1. RECONOCIMIENTO

Esta fase se centra en la búsqueda de un objetivo susceptible de ser atacado. El reconocimiento puede ser de dos tipos: activo, o pasivo. En ambos casos implica la recopilación de información útil. En el pasivo, el hacker recolecta información sobre un objetivo prefijado o potencial con acciones sencillas y de bajo riesgo. En el activo, el hacker busca por la red un blanco, que aún no tiene prefijado, de forma más arriesgada. El hacker dedicará un tiempo a navegar en busca de una víctima propicia aleatoria. En el caso de un auditor, el objetivo suele estar siempre prefijado. El auditor no navega en busca de una víctima, sino que actúa por

[2] www.owasp.org
[3] http://www.isecom.org/research/osstmm.html

encargo específico. Hay un cliente, propietario de un sitio web, que nos contrata para auditar su URL en concreto.

1.6.1.2. MAPEADO

En esta etapa se utiliza la información recolectada en la fase anterior para examinar el blanco de un modo más profundo. Se trata de entender su funcionamiento, determinar si la web es dinámica o estática, el flujo de navegación entre páginas, cómo se mantiene el estado de las sesiones, si tiene parte autenticada o no, etc. En resumen, se identifican las partes de que se compone la aplicación objetivo, se descubre cómo se relacionan estas partes entre sí, y cuál es el comportamiento lógico esperado del usuario, para empezar a pensar, finalmente, dónde y cómo se puede evitar el flujo lógico, gracias a un comportamiento inesperado, que los desarrolladores del software no hayan tenido en cuenta.

1.6.1.3. DESCUBRIMIENTO

Basándonos en todo lo recolectado anteriormente, ahora el foco se dirige a descubrir las incidencias. Si el atacante tiene que crear algún script, suele empezar a hacerlo en esta fase aunque también en la de explotación. Se empieza a explorar la aplicación con cierto tráfico malicioso dirigido a los puntos en los que creemos que puede ser vulnerable. Se empieza también a planear los ataques contra los puntos más débiles de la aplicación o contra la configuración específica que presenta. En ocasiones, en esta fase aún se realiza algo de mapeo. Entran en juego las herramientas propias de ataque y los diccionarios de fuzzing para el testeo de parámetros.

1.6.1.4. EXPLOTACIÓN

Llegamos a la última fase con todo preparado para comenzar los ataques. Un único defecto encontrado en la aplicación puede dar acceso a otras partes de la misma, o incluso a otras máquinas, que previamente no eran vulnerables. Es una fase cíclica, ya que en cada intento, pueden descubrirse más puntos donde repetir el ataque con éxito. Un hacker o auditor inexperto suele saltarse las fases anteriores y empezar por esta directamente.

1.6.2. LISTADO DE ACTIVIDADES A REALIZAR

Tendremos dos conjuntos de actividades a realizar durante el proyecto: el primero, estará compuesto por todas las tareas que forman parte de las cuatro fases del proceso de auditoría, y el segundo, estará compuesto por actividades adicionales necesarias para llevar a cabo el proyecto, en esencia: la documentación y estudio previo, la elaboración de la memoria de todo el trabajo, la creación del informe de auditoría del caso práctico, y la presentación del trabajo y los resultados.

Respecto el primer grupo, cada una de las fases anteriores se compone de un conjunto variable de actividades que se pueden realizar. Cada auditoría es diferente y requiere acciones distintas. En consecuencia, en cada una se harán actividades diferentes según el criterio del profesional que realice el trabajo y en función de las tecnologías o fallos a los que se enfrente.

A continuación se muestra una lista de las tareas más comunes de las cuatro fases mencionadas:

RECONOCIMIENTO

Supone un 10% del total del proceso. Tareas frecuentes:

1) **Definición del alcance**: servidores, URLs, máquinas a auditar, dominios, etc.
2) **Búsquedas en registros de internet**: whois, registros DNS, transferencia de zona, etc.
3) **Consultas en páginas públicas**: listas de correos técnicas, grupos de noticias, páginas de ofertas de empleo, redes sociales, etc. En todas ellas se suele revelar las tecnologías que utiliza la empresa o los campos donde tiene carencias.
4) **Consultas con motores de búsqueda**: Google, Yahoo, etc. Búsqueda de información asociada a los dominios que se quieren auditar.
5) **Búsqueda de sub-dominios**: para encontrar más máquinas dentro de la misma red.
6) **Elaboración de diccionarios**: creación de listados de las palabras que componen la web, para posterior utilización en otros tipos de ataques.

MAPEO

Supone un 20% del total del proceso. Tareas frecuentes:

1) Escaneo de puertos.
2) Versión del sistema operativo y otras aplicaciones.
3) Análisis SSL.
4) Análisis de balanceo de carga y firewalls de aplicaciones web.
5) Análisis de la configuración del software.
6) Spidering. Análisis de los resultados.

DESCUBRIMIENTO

Supone un 45% del total del proceso. Tareas frecuentes:

1) Uso de escáneres automáticos web.
2) Fuzzing de los parámetros de la aplicación.

EXPLOTACIÓN

Supone un 25% del total del proceso. Esencialmente, se trata del lanzamiento de los diferentes ataques. Los ataques concretos dependen de los resultados de las fases anteriores.

1.6.3. ANÁLISIS DE RIESGOS

La realización de la auditoría web no suele presentar ningún riesgo más allá de la sobrecarga del servidor que aloja la página o la interrupción del servicio debido a algún ataque de prueba (por esta causa, se avisa al cliente con antelación del momento exacto en que se va a lanzar un determinado ataque), pero sí pueden darse algunos casos de malas prácticas que podrían implicar riesgos o desembocar en ellos a largo plazo. Algunas de estas situaciones pueden verse en el anexo 1.

1.7. PLANIFICACIÓN TEMPORAL PROPUESTA

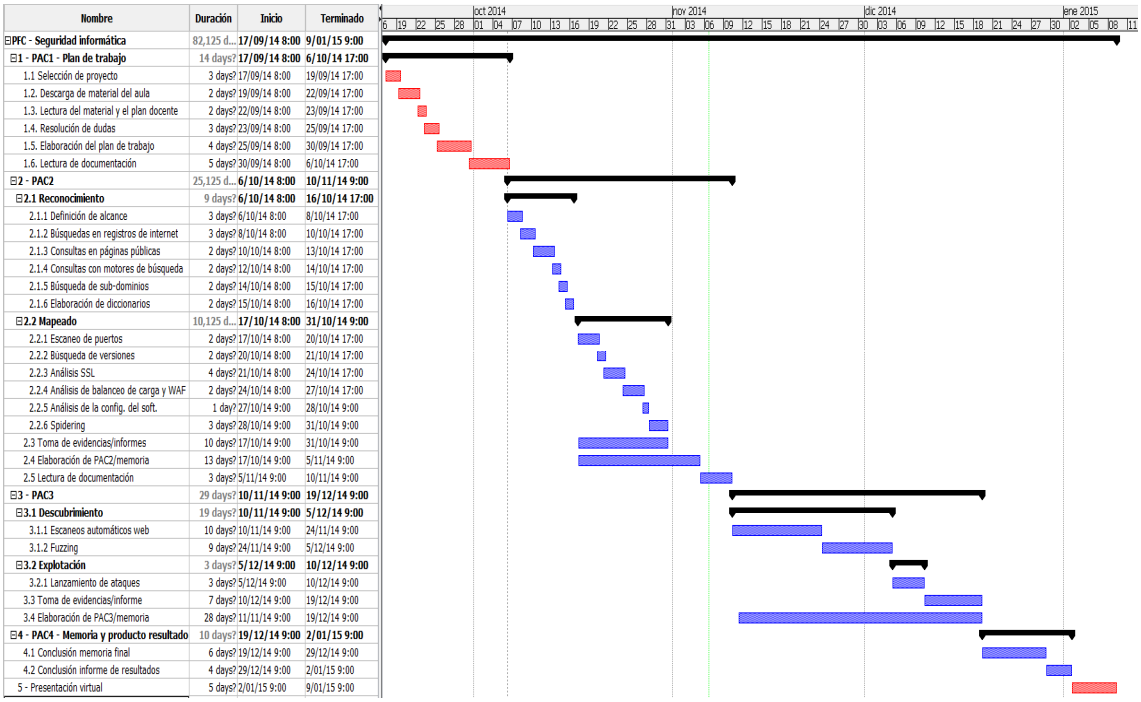
Utilizando la aplicación Open-proj[4] de la web de Sourceforge, vamos a mostrar en un diagrama de Gannt una posible planificación temporal completa para el proyecto que estamos

[4] http://sourceforge.net/projects/openproj/

realizando. En el diagrama se recogerá toda la planificación detallada que se puede consultar en el anexo 2.

Como el programa tiene su propia manera de contar los días laborales entre el 17 de septiembre y el 9 de enero, estas fechas serán una aproximación a las presentadas en las tablas de la planificación detallada.

La propuesta completa final es la siguiente:



Nombre	Duración	Inicio	Terminado
⊟PFC - Seguridad informática	82,125 d...	17/09/14 8:00	9/01/15 9:00
⊟1 - PAC1 - Plan de trabajo	14 days?	17/09/14 8:00	6/10/14 17:00
1.1 Selección de proyecto	3 days?	17/09/14 8:00	19/09/14 17:00
1.2. Descarga de material del aula	2 days?	19/09/14 8:00	22/09/14 17:00
1.3. Lectura del material y el plan docente	2 days?	22/09/14 8:00	23/09/14 17:00
1.4. Resolución de dudas	3 days?	23/09/14 8:00	25/09/14 17:00
1.5. Elaboración del plan de trabajo	4 days?	25/09/14 8:00	30/09/14 17:00
1.6. Lectura de documentación	5 days?	30/09/14 8:00	6/10/14 17:00
⊟2 - PAC2	25,125 d...	6/10/14 8:00	10/11/14 9:00
⊟2.1 Reconocimiento	9 days?	6/10/14 8:00	16/10/14 17:00
2.1.1 Definición de alcance	3 days?	6/10/14 8:00	8/10/14 17:00
2.1.2 Búsquedas en registros de internet	3 days?	8/10/14 8:00	10/10/14 17:00
2.1.3 Consultas en páginas públicas	2 days?	10/10/14 8:00	13/10/14 17:00
2.1.4 Consultas con motores de búsqueda	2 days?	12/10/14 8:00	14/10/14 17:00
2.1.5 Búsqueda de sub-dominios	2 days?	14/10/14 8:00	15/10/14 17:00
2.1.6 Elaboración de diccionarios	2 days?	15/10/14 8:00	16/10/14 17:00
⊟2.2 Mapeado	10,125 d...	17/10/14 8:00	31/10/14 9:00
2.2.1 Escaneo de puertos	2 days?	17/10/14 8:00	20/10/14 17:00
2.2.2 Búsqueda de versiones	2 days?	20/10/14 8:00	21/10/14 17:00
2.2.3 Análisis SSL	4 days?	21/10/14 8:00	24/10/14 17:00
2.2.4 Análisis de balanceo de carga y WAF	2 days?	24/10/14 8:00	27/10/14 17:00
2.2.5 Análisis de la config. del soft.	1 day?	27/10/14 9:00	28/10/14 9:00
2.2.6 Spidering	3 days?	28/10/14 9:00	31/10/14 9:00
2.3 Toma de evidencias/informes	10 days?	17/10/14 9:00	31/10/14 9:00
2.4 Elaboración de PAC2/memoria	13 days?	17/10/14 9:00	5/11/14 9:00
2.5 Lectura de documentación	3 days?	5/11/14 9:00	10/11/14 9:00
⊟3 - PAC3	29 days?	10/11/14 9:00	19/12/14 9:00
⊟3.1 Descubrimiento	19 days?	10/11/14 9:00	5/12/14 9:00
3.1.1 Escaneos automáticos web	10 days?	10/11/14 9:00	24/11/14 9:00
3.1.2 Fuzzing	9 days?	24/11/14 9:00	5/12/14 9:00
⊟3.2 Explotación	3 days?	5/12/14 9:00	10/12/14 9:00
3.2.1 Lanzamiento de ataques	3 days?	5/12/14 9:00	10/12/14 9:00
3.3 Toma de evidencias/informe	7 days?	10/12/14 9:00	19/12/14 9:00
3.4 Elaboración de PAC3/memoria	28 days?	11/11/14 9:00	19/12/14 9:00
⊟4 - PAC4 - Memoria y producto resultado	10 days?	19/12/14 9:00	2/01/15 9:00
4.1 Conclusión memoria final	6 days?	19/12/14 9:00	29/12/14 9:00
4.2 Conclusión informe de resultados	4 days?	29/12/14 9:00	2/01/15 9:00
5 - Presentación virtual	5 days?	2/01/15 9:00	9/01/15 9:00

Figura 1: Diagrama de Gannt del proyecto completo

1.8. ORGANIZACIÓN DE LA MEMORIA

Organizaremos el presente trabajo en varios capítulos consecutivos en los que iremos desarrollando los objetivos planteados:

1. **Introducción**. Este capítulo recoge, principalmente, cuáles son los objetivos del proyecto, el contexto en que se desarrolla, los recursos necesarios para llevarlo a cabo, las fases que se implementarán, el listado de tareas a desempeñar en cada fase, y la planificación temporal que se seguirá.

2. **Situación actual y metodologías de partida**. En este capítulo se explicará cuáles son las vulnerabilidades y riesgos más comunes en la actualidad y cómo, las metodologías de partida en las que nos basamos, categorizan éstas vulnerabilidades y organizan los casos de pruebas.

3. **La auditoría web profesional. Aspectos**. Es un capítulo que trata diversos aspectos de importancia en la práctica de la auditoría en un entorno empresarial, como por ejemplo, aspectos contractuales, normativos, y legales, que el auditor debe conocer. También recoge un conjunto de buenas prácticas y hábitos en el trabajo, repasa el perfil del

auditor, su ética profesional, y explica la elaboración y estructura del informe de resultados de una auditoría.

4. **Metodología. Fase de reconocimiento**. El capítulo desarrolla toda la fase de reconocimiento de una auditoría web a través de un caso práctico real.

5. **Metodología. Fase de mapeado**. El capítulo desarrolla la fase de mapeado de una auditoría web a través de un caso práctico real.

6. **Metodología. Fases de descubrimiento y explotación**. En un solo capítulo se desarrollan conjuntamente las dos últimas fases de la metodología propuesta, ya que están estrechamente relacionadas y sus actividades tienden a solaparse. Se mostrarán, de nuevo, a través del caso práctico real que nos ocupa.

7. **Conclusiones**. En este capítulo exponemos cómo hemos cumplido con los objetivos del proyecto, y presentamos las conclusiones a las que hemos llegado al finalizar el proceso.

8. **Glosario, bibliografía y anexos**. Estas tres secciones finales nos aportan: un diccionario de términos propios del campo de estudio, las referencias consultadas para documentar el proyecto y los sitios web donde encontrar las herramientas utilizadas para su desarrollo, y un conjunto de anexos.

CAPÍTULO 2. SITUACIÓN ACTUAL Y METODOLOGÍAS DE PARTIDA

La seguridad de las aplicaciones ha pasado en pocos años de ser algo accesorio a ser algo de vital importancia para el buen funcionamiento de toda aplicación conectada a la red. Con el surgimiento de Internet, también surgió una nueva modalidad de delito: el delito informático, que no son más que los delitos de siempre (robo, espionaje, estafa, etc), pero perpetrados a través de la red contra aplicaciones informáticas, aprovechando los fallos de programación, funcionamiento, o configuración que puedan tener, o lo que es lo mismo, los agujeros de seguridad [3]. A la persona que es capaz de descubrir estos agujeros de seguridad para atacar a un sistema y entrar en él sin permiso legítimo, se la llama "hacker".

El perfil del "hacker" no siempre consiste en una única persona. En ocasiones, puede tratarse de un grupo clandestino o mafia organizada, que busca lucrarse perpetrando robos, no de dinero en exclusiva, sino principalmente de datos confidenciales o información. Estos datos, una vez obtenidos, pueden ser vendidos a empresas que son competencia directa de la empresa atacada (lo que entendemos como espionaje industrial). Un ejemplo muy habitual es el robo de patentes, alojadas en servidores, entre empresas de un mismo sector y competidoras entre sí. Otro ejemplo puede ser el de obtener información de los clientes y de las ofertas presentadas por parte de la competencia.

El atacante también puede ser un técnico muy experimentado, con grandes conocimientos en lenguajes de programación, que ha creado un malware (un virus, gusano, troyano, o cualquier programa dañino similar), y lo que busca es propagarlo por la red. A veces, estos virus se limitan a dañar los ordenadores infectados, por ejemplo, provocando el borrado de datos o corrompiéndolos, pero la mayoría de veces el objetivo que se busca es el desvío de los recursos del ordenador infectado para el uso del atacante. Sin que el propietario lo sepa, el virus se instala en su sistema, y cada vez que se conecta, éste consume los recursos del ordenador para ejecutar tareas propias del dueño del malware. En otras ocasiones, los atacantes cifran datos del servidor legítimo y exigen un rescate al propietario, para recuperar la información. En cierto modo es un "secuestro" de los datos; al software empleado para estas prácticas se lo conoce como "ransomware".

La actividad de hacking [4] no se limita a personas o mafias; también los países producen actividad hacker dirigida a atacar empresas de otros países, o al espionaje de las comunicaciones de estos[5]. Según las estadísticas, a la cabeza de estas prácticas estarían países como China, Rumanía, Rusia o Estados Unidos.

Como vemos, hay muchos peligros en la red, y la tendencia es que la actividad hacker aumente en los próximos años. Esta situación es la que el auditor trata de contrarrestar con su trabajo. Comparativamente, la figura del auditor y la del hacker son similares, pero les diferencia la buena o mala intencionalidad de sus objetivos y motivaciones. Es por eso que se usa la expresión "hacker ético" o sombrero blanco (White hat), para definir al auditor, y simplemente hacker, o sombrero negro (Black Hat) para definir al intruso ilegítimo.

En línea con la intención de contrarrestar los peligros de la red mediante auditorías técnicas de seguridad, se han elaborado, durante los últimos años, metodologías que ayudan a

[5] Ejemplo de noticias relacionadas: http://www.20minutos.es/noticia/1843571/0/espionaje/eeuu/privacidad/

la realización de este tipo de proyectos. Estas metodologías se actualizan periódicamente por un conjunto de voluntarios para mantenerlas al día. A continuación vamos a exponer, de forma breve, la última versión de los casos de pruebas catalogados por OSSTMM, y la última versión de las categorías de vulnerabilidades identificadas por OWASP [11, 12].

2.1. OSSTMM

El manual OSSTMM define todos los conceptos básicos del proceso de auditoría de sistemas y redes. Su objetivo es crear un estándar para evaluar la seguridad de la información almacenada digitalmente.

Los sistemas, a menudo, manejan datos sensibles. Los datos pueden pertenecer a la entidad propietaria del sistema o a terceros, por ejemplo, clientes que hagan uso del servicio. Estos datos deben ser protegidos adecuadamente y, si aplica, cumplir con las leyes de Protección de Datos. Una entidad que desee ofrecer un servicio en internet no puede ser descuidada en lo referente al blindaje de sus aplicaciones; serlo podría valerle una pérdida de su reputación, y en consecuencia, pérdida de clientes. Por este motivo, todo sistema tiene la necesidad de velar por la confidencialidad, la integridad y la disponibilidad de los datos que maneja. Estos tres conceptos son los tres pilares sobre los que se apoya la seguridad informática:

- **Confidencialidad**: se define como la cualidad que garantiza que la información es accesible sólo para aquellos autorizados a tener acceso, o dicho de otro modo, se refiere a evitar que acceda a la información privada quien no tiene permiso para hacerlo.

- **Integridad**: es la cualidad que garantiza que los datos serán verdaderos y fiables, ya que sólo se podrán modificar por el propietario legítimo de los mismos, o por aquel con permiso para su manipulación (por ejemplo, un administrador).

- **Disponibilidad**: es la característica que debe garantizar que los servicios ofertados o los datos a consultar van a ser accesibles en cualquier instante por las personas autorizadas que lo requieran.

Una auditoria web debe ayudar a que la web o servicio auditado cumpla con estas tres propiedades de la seguridad de la información [5].

2.1.1. CASOS DE PRUEBA

OSSTMM agrupa los tipos de pruebas [6] que se pueden hacer sobre las aplicaciones, en las siguientes cinco categorías [7]:

2.1.1.1. PRUEBAS DE SEGURIDAD HUMANA

Este conjunto de pruebas son de tipo psicológico, y va dirigido al personal que guarda, protege o vigila los bienes o activos de la entidad puesta a prueba, y más en general, a todo el que esté directamente involucrado con el objetivo del ataque. Requiere de cierto trato o interacción con estas personas, que conocen los datos o tienen acceso a ellos, con el propósito de engañarlas para que nos revelen información, o la forma de acceder a la misma. Se basa, por tanto, en encontrar las vulnerabilidades de las personas, en lugar de las máquinas. Esto se conoce como "ingeniería social".

La utilidad de este grupo de pruebas es comprobar el grado de concienciación del personal respecto de la seguridad, y ver si cumplen con lo indicado en la política de la empresa.

Por ejemplo, si la política indica que hay que cambiar la contraseña del ordenador cada tres meses, y no apuntarla en un "post-it" a la vista, en este grupo se comprobaría si los empleados siguen la recomendación. También se comprueba si hay algún tipo de vacío o falta en las normas estándar de seguridad que pueda ser aprovechado por gente externa a la empresa.

Sobre el grupo puesto a prueba, hay que tener en cuenta que para que la gente sepa salvaguardar la seguridad adecuadamente, antes debe ser entrenada, concienciada, e informada de cuáles son las responsabilidades asociadas a su puesto respecto a la privacidad de los datos que maneja. Según el grado de privilegios al que puede acceder por su puesto de trabajo, la exigencia en este ámbito será distinta.

Además, las pruebas deben hacerse por un auditor autorizado, y deben practicarse sin el conocimiento del personal objetivo. Lógicamente, si saben que van a ser puestos a prueba, estarán más alerta que de costumbre a la espera de cualquier suceso "extraño", y su comportamiento no será genuino, sino que variará del que tendrían en un día normal y corriente. En el caso de la auditoría de aplicaciones web, estas pruebas no suelen estar incluidas.

2.1.1.2. PRUEBAS DE SEGURIDAD FÍSICA

En esta clasificación entra todo lo concerniente a la seguridad material, o física, del entorno. Al contrario que en el caso anterior, aquí no es necesaria ninguna comunicación con el personal involucrado. Sin embargo, sí hace falta que haya proximidad física entre el auditor y el objeto material que se pretende atacar. La finalidad de este grupo de pruebas es comprobar que las defensas y barreras, lógicas o físicas, que protegen los activos, funcionan correctamente, e impiden todo acceso ilegítimo. Además se debe verificar que hay servicios de protección contra incendios, tales como extintores y detectores de humo. Un ejemplo sería comprobar que el auditor no puede ganar acceso al CPD de la empresa auditada, con intención de desconectar los servidores, borrarlos o romperlos, sin disponer de la llave de la puerta, la clave, tarjeta, o el permiso oportuno para traspasar la barrera de entrada. Otro ejemplo podría ser apagar el aire del CPD para ver si lo detecta el sistema de monitorización.

La utilidad de estas pruebas también está en detectar algún vacío o brecha en el cumplimiento de las normas de seguridad indicadas en la política de la empresa, o en las regulaciones o legislación pertinente. Este grupo tampoco suele aplicarse en el caso concreto de la auditoría web.

2.1.1.3. PRUEBAS DE SEGURIDAD INALÁMBRICA

Las redes inalámbricas envían datos por medio de ondas electromagnéticas. Estas ondas viajan por el aire, por lo que es más fácil ganar acceso a ellas que a las redes cableadas. Este grupo de pruebas estudia la seguridad de los dispositivos electrónicos inalámbricos que comunican con los ordenadores por medio de emisiones de señales, como por ejemplo los enrutadores "wifi". Se trata de verificar las medidas desplegadas para denegar los accesos no autorizados a la información que pueda ser interceptada en las ondas electromagnéticas; verificar, también, las medidas que protegen las comunicaciones sin cables de las interferencias del entorno que podrían provocar denegación de servicio; y por último, verificar las medidas para prevenir transmisiones desde la máquina conectada, cuya información podría ser interceptada durante su procesamiento, sin conocimiento del personal técnico.

El auditor analizará el tráfico de la red para determinar qué algoritmos de cifrado se están empleando, o si la red funciona en abierto. De la misma forma, analizará el esquema de autenticación de la red. Una tarea común puede ser capturar el tráfico de la red "wifi", y después intentar obtener la clave secreta de acceso a la red. Si logra conectarse y hacer uso de ella, se considera que las pruebas han tenido éxito.

2.1.1.4. PRUEBAS DE SEGURIDAD DE LAS TELECOMUNICACIONES

Este apartado va dirigido a las comunicaciones tradicionales que viajan por medio de los cables telefónicos. De este modo se separan las pruebas para las señales electromagnéticas del apartado anterior, de las pruebas para la red telefónica.

El auditor debe disponer de equipamiento para hacer pruebas tanto de tipo digital como analógico, ya que puede encontrarse con múltiples tipos de dispositivos. Entre las tareas más frecuentes está la verificación de la configuración, que consiste, por ejemplo, en comprobar si las listas de acceso están bien diseñadas, analizar los servicios disponibles en la PBX y ver si son vulnerables, buscar módems que no se usen y que estén a la escucha, comprobar que el nivel de parcheado del fax es seguro, etc.

2.1.1.5. PRUEBAS DE SEGURIDAD DE LAS REDES DE DATOS

Este grupo es similar al anterior pero dirigido a las redes de comunicaciones entre ordenadores, a través de cables de red. Las pruebas comprueban los controles de acceso a la información que circula por las redes, y la existencia de vulnerabilidades en los sistemas interconectados. En caso de que el auditor dé con sistemas vulnerables, pedirá permiso al cliente antes de explotarlos, como medida de prevención.

En este caso, el acceso físico a los sistemas no es necesario, ni tampoco ningún tipo de interacción con el personal. Se puede realizar la batería de pruebas a distancia, desde cualquier otra ubicación, siempre que se disponga de un medio con acceso a la red.

Un ejemplo sería verificar la política de contraseñas de un dominio: si se exige que las contraseñas se cambien de forma periódica, si se exige una longitud determinada, si la cuenta se bloquea tras varios intentos fallidos de acceso, o si se usan caracteres especiales.

2.2. OWASP

La guía de pruebas OWASP no abarca tantos tipos de pruebas como hemos visto en el punto anterior, sino que se centra expresamente en la seguridad de las aplicaciones web [8]. Su objetivo principal es conseguir que las empresas de desarrollo del software incluyan la comprobación de seguridad como parte de su proceso estándar de desarrollo, dando como resultado un software seguro. La comunidad cuenta con muchos colaboradores que se encargan de mantener la guía actualizada, gracias a lo cual, evoluciona al mismo ritmo que las amenazas. A continuación, exponemos la categorización que hace OWASP de los ataques más comunes que podemos encontrar hoy.

2.2.1. OWASP TOP TEN

La comunidad OWASP recoge en una lista los diez riesgos, o tipos de ataques, más críticos en la seguridad de las aplicaciones web, lista que actualiza casi todos los años para mantenerla al día.

El top [9, 10] en la actualidad se compone de los siguientes tipos de ataques:

A1: INYECCIÓN

Las inyecciones de código (SQL, OS, o LDAP), ocurren cuando se envía datos no confiables a la aplicación, como parte de un comando o una consulta. El código malicioso que introduce el atacante puede confundir a la aplicación, que acaba ejecutando ese código por error, y permitiendo el acceso no autorizado a bases de datos, o la ejecución de comandos.

A2: FALLO DE AUTENTICACIÓN Y GESTIÓN DE SESIONES

Las funciones que controlan el acceso autenticado y la gestión de sesiones, a menudo no se implementan correctamente. Como resultado, frente a un ataque, las claves y contraseñas pueden quedar comprometidas, y también los identificadores de sesión, con lo que el atacante podría apropiarse de la identidad de un usuario legítimo y suplantarle.

A3: CROSS-SITE SCRIPTING (XSS)

Ocurre cuando una aplicación confía en los datos suministrados por el usuario, y devuelve al navegador web código de scripting sin haberlo validado o filtrado apropiadamente. Esto permite al atacante ejecutar código en el navegador de la víctima, con lo que puede capturar sus sesiones de usuario y redirigir su navegación hacia sitios web que la víctima no desea visitar, por ejemplo, sitios publicitarios, o que contengan malware.

A4: REFERENCIAS INSEGURAS A OBJETOS

Este error ocurre cuando se permite a un atacante modificar el identificador que referencia a un objeto, sin verificar si está autorizado a acceder a este objeto. Esto hace posible acceder a documentación sensible, como puede ser un directorio, un fichero, o tablas de la base de datos, sin disponer de la autorización necesaria, saltando el control de accesos.

A5: FALLO DE CONFIGURACIÓN

El atacante puede acceder a información sensible si la configuración del objetivo del ataque no es la correcta, por ejemplo, un servidor, aplicación, navegador, base de datos, o plataforma. La configuración de todos estos objetos debe estar definida, implementada y desplegada, y las versiones de software deben estar actualizadas. Las configuraciones por defecto de algunos dispositivos suelen ser inseguras y los hackers lo saben. Por ejemplo, ciertos enrutadores llevan de inicio como usuario y clave la palabra “admin”.

A6: EXPOSICIÓN DE DATOS SENSIBLES

Las aplicaciones web deben proteger los datos sensibles que un usuario introduzca para operar, por ejemplo, datos de tarjetas de crédito al realizar una compra, o contraseñas. Estos datos deben enviarse cifrados para que, en caso de ser interceptados, no puedan ser conocidos ni modificados.

A7: AUSENCIA DE CONTROL DE ACCESO A FUNCIONES

Las aplicaciones web verifican los derechos de acceso, a nivel de función, antes de hacer visible la función en la interfaz del usuario. Sin embargo, también deben controlar el acceso en el servidor al acceder a cada función. Si estas peticiones de acceso no se verifican, un atacante podría crear peticiones para acceder a las funciones sin autorización.

A8: CROSS-SITE REQUEST FORGERY (CSRF)

Estos ataques consisten en hacer que un usuario víctima, autenticado en una web objetivo, envíe peticiones HTTP falsas con la intención de forzarlo a efectuar una operación sin su consentimiento. Estas peticiones incluyen la sesión del usuario y cualquier otra información de autenticación, ya que tienen una sesión válida y por lo tanto la web vulnerable las acepta como peticiones legítimas de la víctima. Este tipo de ataque requiere utilizar un servidor controlado por el atacante que sea visitado por la víctima mientras está autenticado en la web objetivo.

A9: COMPONENTES CON VULNERABILIDADES CONOCIDAS

Hay módulos de software, como librerías, que por defecto funcionan con todos los privilegios activados. Si fueran atacados y resultaran ser vulnerables, podrían permitir con facilidad una intrusión en el servidor, o incluso un borrado de datos severo. Las defensas de las aplicaciones se debilitan si utilizan estos componentes con vulnerabilidades conocidas, además, el rango de ataques posibles se eleva. Por eso, no es recomendable el uso de estos componentes.

A10: REDIRECCIONES Y REENVÍOS NO VALIDADOS

Es habitual que las aplicaciones web redirijan y reenvíen a los usuarios hacia otras páginas web, y que para ello, usen datos no confiables con los que determinan las páginas de destino. Sin la validación apropiada al hacer estas redirecciones, un atacante podría redirigir al usuario a sitios web no fiables, como webs que contengan malware, o podría usar la redirección para ganar acceso no autorizado a ciertas páginas.

2.2.2. VULNERABILIDADES WEB TÍPICAS

Aparte de los tipos de ataques vistos en el top ten, existen otros tipos de vulnerabilidades web, muy típicos y comunes hoy, que conviene conocer para desempeñar adecuadamente la labor de auditoría web. Son estos:

2.2.2.1. ESCALADA DE PRIVILEGIOS

Consiste en verificar que un usuario no puede modificar sus privilegios o su perfil, dentro de la aplicación. Si pudiera, un atacante podría registrarse como usuario, para después realizar ataques que le hagan ganar privilegios, por ejemplo, de administrador.

2.2.2.2. INCLUSIÓN DE FICHEROS (LFI/RFI)

Consiste en obtener ficheros ajenos a la aplicación, ya sea de forma local (LFI) o de forma remota (RFI). “Local file inclusion" permite al atacante leer ficheros que están presentes en el mismo servidor de la aplicación, y “Remote file inclusion" permite al atacante obtener ficheros desde un servidor remoto. Estos ataques pueden utilizarse para leer ficheros con información sensible (como el `/etc/passwd`) o para ejecutar código de scripting (javascript, PHP, etc) en el servidor web vulnerable.

2.2.2.3. ENUMERACIÓN DE USUARIOS

Consiste en encontrar y recolectar un conjunto de usuarios válidos en el sistema al observar las respuestas ante intentos de autenticación fallidos. En algunos casos, la aplicación responde indicando que la contraseña o el usuario son incorrectos. Cuando esto ocurre es posible determinar los usuarios existentes. El atacante podría, más tarde, realizar ataques de fuerza bruta contra estos usuarios encontrados, para ver si logra hacerse con alguna clave que le permita el acceso. Por ejemplo, se puede lograr esta información cuando intentamos hacer un registro en una web, y ésta nos responde que ese nombre de usuario ya existe.

2.2.2.4. ATAQUES POR FUERZA BRUTA

Consiste en encontrar contraseñas válidas de acceso a base de probar todas las posibilidades contra la aplicación, de forma sistemática, e ir descartándolas una a una hasta dar con la correcta. En concreto, en aplicaciones web, donde el acceso se gana por usuario y contraseña, se comprueban empleando un diccionario todas las combinaciones posibles de letras, números y caracteres, que se pueden imprimir con un teclado. En caso de enfrentarnos a un móvil o teclado numérico, lo que probaríamos sería todas las combinaciones posibles de “pin”, desde el 0000, hasta el 9999.

CAPÍTULO 3. LA AUDITORÍA WEB PROFESIONAL. ASPECTOS.

En esta sección vamos a explicar cuáles son los aspectos más destacados de la práctica de la auditoría en un entorno empresarial.

3.1. PERFIL DE AUDITOR

El auditor informático es un perfil principalmente técnico, de grado elevado. Su trabajo no consiste en un proceso fijo y repetitivo, que se pueda automatizar, sino que tiene una gran componente creativa. Debido a que cada proyecto que realice será distinto de los anteriores, y cada empresa a la que audite se dedicará a una actividad distinta, o a un sector distinto, el auditor debe tener la habilidad de aplicar sus conocimientos a múltiples situaciones diferentes.

Su trabajo requiere, además del estudio continuo de la materia, cierto grado de investigación para encontrar problemas, hallar la forma de explotarlos, y ofrecer después recomendaciones efectivas para resolverlos.

La calidad de una auditoría dependerá, en gran medida, de la profesionalidad, la destreza, y la experiencia en campo del auditor. En él recae la responsabilidad de lograr unos buenos resultados, y de que el proceso de auditoría resulte rentable, y de gran utilidad, para la empresa que la solicita. Para lograr estos buenos resultados, el auditor debe contar también con un nivel alto de comprensión de aquellas tecnologías que va a revisar. Esto conlleva que la persona no sólo tenga una formación académica de grado superior, como una ingeniería, sino también una formación complementaria que le mantenga al día y con la que renueve y actualice sus conocimientos, por ejemplo, a través de talleres, conferencias, seminarios, "papers", certificaciones oficiales, cursos, etc.

Para profundizar más en las capacidades o habilidades deseables de un perfil auditor, podemos consultar el anexo 3.

3.2. TIPOS DE TEST

La auditoría puede ser de dos tipos: interna o externa. La interna es la que el auditor realiza sobre la propia organización que le ha contratado y para la que trabaja. Forma parte del equipo, por lo que conocerá con profundidad el funcionamiento de la empresa y al personal. La externa es la que se realiza desde fuera de la organización, a través de la red, sobre una empresa cliente ajena al auditor, y con la que este no tiene relación ninguna. Además, existen tres tipos de enfoques al realizar los test de intrusión [8]:

3.2.1. ENFOQUE DE CAJA NEGRA

Se trata de analizar una aplicación que se ejecuta remotamente sin conocer nada sobre su funcionamiento interno, para hallar agujeros de seguridad. Se parte del punto de vista de un usuario cualquiera y se realiza totalmente a ciegas, sin disponer más que de información pública como la dirección URL o el nombre del dominio. Se simulan los métodos del atacante y se utilizan sus mismos recursos. Así, se puede evaluar el nivel de riesgo real al que se expone el cliente bajo la premisa de que un "hacker" encontrará lo mismo que encuentre el auditor. Por ejemplo, las vulnerabilidades de "Cross-Site Scripting" o de inyección de código son técnicas de caja negra. También lo son la gestión de contraseñas y, lógicamente, las técnicas de fuerza bruta.

3.2.2. ENFOQUE DE CAJA BLANCA

Se trata del caso contrario al de caja negra. El auditor no actúa a ciegas, sino que posee de entrada conocimientos exhaustivos sobre el sistema. El cliente provee al auditor con toda la información necesaria para que la revisión sea más profunda, minuciosa, y dirigida a los elementos de mayor interés para él. Así, el auditor puede conocer la configuración y topología del sistema, ver diagramas de red, listas de servidores, el código fuente de la aplicación, y otra documentación útil. Este tipo de test no sólo evalúa las amenazas directas, sino también las potenciales. Permite al auditor corregir secciones de código, corregir configuraciones potencialmente peligrosas, detectar puertas traseras, o asesorar con mayor precisión para reparar los fallos en el sistema y evitar los defectos explotables. Por ejemplo, podemos aplicar las técnicas de caja blanca a la vulnerabilidades de gestión de sesiones, fallos de autenticación y configuración, referencias inseguras a objetos, o ausencia de control de acceso a funciones.

3.2.3. ENFOQUE DE CAJA GRIS

Este enfoque es un punto intermedio entre los dos tipos anteriores. Se realizan ataques reales, como ocurre en caja negra, pero a la vez, existe algún conocimiento parcial de la aplicación o sistema, como en caja blanca, y si el auditor lo considera oportuno, puede solicitar más información técnica [12].

3.3. ASPECTOS CONTRACTUALES DE LA AUDITORÍA

Además de los aspectos técnicos, la auditoría tiene una serie de aspectos contractuales, que son aquellos que el auditor firma con la empresa cliente que lo contrata, en cada ocasión en que éste entra a formar parte de un proyecto. En este apartado vamos a conocer los aspectos más comunes:

3.3.1. PERMISO EXPLÍCITO

La realización de las pruebas debe ser aprobada por el cliente, pero también por todos los propietarios del sistema involucrados en la auditoría. El auditor debe conseguir el permiso explícito de cada propietario implicado, firmado por escrito. Por ejemplo, si la aplicación del cliente está ubicada en un servidor propiedad de otra empresa, ésta deberá dar también su consentimiento al auditor para proceder a realizar las pruebas.

3.3.2. ACUERDO DE NO DIVULGACIÓN

En muchas profesiones es necesario firmar un acuerdo de no divulgación entre las entidades que toman parte de un negocio con el fin de asegurar la confidencialidad de la información sensible o valiosa que se comparte. La consultoría informática es una de esas profesiones. Durante la auditoría, el auditor accederá a gran cantidad de datos clasificados, que serán de mucho valor para la entidad que contrata el servicio. Más aún, su trabajo consiste, en parte, en intentar hacerse con estos datos aunque no le sean proporcionados directamente. El acuerdo hace posible que ambas partes, la empresa que contrata una auditoría y el auditor que la realiza, queden protegidos y puedan confiar en que se actuará con la discreción apropiada. Igual que en el caso del permiso explícito, es importante que el acuerdo de no divulgación se haga por escrito, y que describa el ámbito en que afecta, y cómo debe ser usada la información compartida.

3.3.3. DEFINICIÓN DEL ALCANCE DEL PROYECTO

Consiste en definir, con concreción, todas aquellas actividades que deben realizarse durante el proyecto. Sobre ésta definición se estimará la duración del proyecto, el personal dedicado a él, y el coste

que alcanzará. Además de las actividades, se definirán el número de máquinas a analizar, las URLs, los objetos, los tipos de pruebas que se harán, etc. Es el primer paso en el proceso de auditoría. En el siguiente capítulo se hablará más a fondo del alcance, como parte de la fase de reconocimiento.

3.3.4. REGLAS DE COMPROMISO

Consiste en definir todas aquellas tareas y pruebas que no hay que realizar y por tanto quedan fuera de la auditoría. Estas tareas "prohibidas" pueden especificarse también en el alcance. Las razones para definir lo que no debe hacerse son muchas, por ejemplo, que el auditor no malgaste el tiempo en auditar algo que el cliente no desea que se realice. En general, cuando la aplicación está en producción, se prohíben los ataques de denegación de servicio. En las reglas también se suele especificar el horario de las pruebas. Se puede acordar que el auditor notifique las vulnerabilidades críticas tan pronto como las encuentre, en lugar de esperar a la entrega del informe de resultados al concluir el proyecto. Si se desean incluir ataques de ingeniería social (pruebas sobre la psicología del personal de la empresa), es importante definir con claridad los límites de cada prueba que vaya a hacerse, y todo lo que no puede hacerse.

3.3.5. AUSENCIA DE GARANTÍA

Debido a que el tiempo de la auditoría está muy acotado, es importante para la empresa auditora añadir una cláusula en sus contratos que especifiquen que la organización no ofrece garantía alguna respecto a haber encontrado todas las vulnerabilidades existentes en la aplicación del cliente. Pese a sus hallazgos, la web podría no ser segura al 100%, por lo que no se hará responsable de posibles ataques en el futuro, ni de las consecuencias de estos, tanto en el caso de que el atacante explote una vulnerabilidad encontrada por la empresa auditora, como en el caso de que explote una vulnerabilidad no encontrada. Si no se firmara ésta cláusula, la empresa auditora podría verse constantemente demandada por sus clientes, que podrían intentar responsabilizarla de sus pérdidas incluso en los casos en que no hayan aplicado las recomendaciones dadas.

3.4. ASPECTOS NORMATIVOS Y LEGALES

Durante la realización de las auditorías, es posible que encontremos vulnerabilidades que afecten a datos personales o a datos de tarjetas de crédito. En estos casos puede haber repercusiones serias, por infringir ciertas leyes como la LOPD, o normativas de obligado cumplimiento como la PCI DSS. Si la empresa no protege bien los datos puede estar incurriendo, aunque sea sin saberlo, en una falta o un delito. El auditor deberá conocer qué hacer en estos casos. Es más, toda vulnerabilidad que encontremos en una aplicación web tendrá un impacto más grave si ésta almacena datos personales o datos de tarjetas de crédito o débito.

3.4.1. LOPD

La Ley Orgánica de Protección de Datos (LOPD)[6] es una ley que tiene por objeto garantizar y proteger los datos personales de las personas. Se fundamenta en el artículo 18 de la Constitución Española de 1978 que habla del derecho a la intimidad.

A partir de esta ley se creó la Agencia Española de Protección de Datos (AEPD)[7] que vela por el cumplimiento de esta normativa. La agencia tiene potestad para sancionar a las

[6] http://www.lopd-proteccion-datos.com/ley-proteccion-datos.php

empresas que incumplan la normativa, ya sea por almacenar datos sin el consentimiento de los usuarios o por protegerlos indebidamente.

Como auditores, hay que tener presente esta normativa para avisar al cliente en caso de que al auditar aplicaciones web encontremos vulnerabilidades que puedan provocar la divulgación de datos personales. Las multas oscilan entre los 600 y los 600.000 euros. Saber guiar al cliente al respecto de cómo cumplir con la LOPD es otra más de las responsabilidades del auditor[8].

3.4.2. PCI DSS

La normativa PCI DSS[9] ("Payment Card Industry Data Security Standard") es un estándar de seguridad de los datos para la industria de tarjetas de crédito. El estándar fue desarrollado por un comité conocido como PCI SSC[10] ("Payment Card Industry Security Standards Council") que está formado por las empresas de tarjetas más importantes del mundo. El objetivo es el de ayudar a las organizaciones que procesen, almacenen o transmitan información relativa a tarjetas de crédito a prevenir los fraudes con tarjetas bancarias.

La normativa PCI DSS es de obligado cumplimiento para toda empresa que procese, guarde o transmita información de tarjetas de crédito o débito. Las empresas que incumplan el estándar se arriesgan a enfrentar auditorías rigurosas, a perder el permiso para procesar pagos con tarjetas o a enfrentarse a cuantiosas multas.

Como auditores, hay que tener presente ésta normativa para incrementar el riesgo de forma apropiada en caso de que, al auditar aplicaciones web, encontremos vulnerabilidades que puedan provocar la divulgación de datos de tarjetas de crédito o débito.

3.5. ÉTICA PROFESIONAL

Ya hemos comentado lo importante que es para el auditor tener una ética profesional. Ser fiel a ésta ética le protegerá de cualquier eventualidad sucedida en relación al desempeño de su trabajo, como problemas legales respecto a su responsabilidad y participación en un proyecto. Mantener una reputación intachable es vital en una profesión que requiere conservar la confianza de cada cliente. Si el auditor pierde la reputación por faltar a su compromiso, perderá también el trabajo y hasta la posibilidad de volver a trabajar en el sector.

Para profundizar más en los aspectos de la profesión, en un conjunto de buenas prácticas y hábitos de trabajo, podemos consultar el anexo 4.

Respecto a la ética, hay dos puntos destacables a tener en cuenta:

3.5.1. MANTENER LA CONFIDENCIALIDAD

Las auditorías se contratan gracias a que este aspecto se mantiene. Si no ocurriera así, la reputación de la profesión se vería afectada y nadie contrataría unos servicios que suponen un riesgo. El

[7] http://www.agpd.es/portalwebAGPD/canaldocumentacion/informes_juridicos/reglamento_lopd/index-ides-idphp.php
[8] https://www.agpd.es/portalwebAGPD/jornadas/dia_proteccion_2011/responsable/index-ides-idphp.php
[9] http://es.wikipedia.org/wiki/PCI_DSS
[10] https://www.pcisecuritystandards.org/security_standards/index.php

auditor no sólo debe guardar el secreto sobre los datos a los que acceda, sino también respecto a las vulnerabilidades, o debilidades que descubra en la empresa a la que audita. Tampoco debe divulgar con qué clientes ha trabajado. La discreción debe mantenerse en el ámbito privado además del laboral, lo que quiere decir que el auditor no puede contarle a sus amigos o familia lo que averigüe sobre sus clientes. Como es lógico, la duración de esta obligación es permanente, no se limita a la duración del contrato. En este sentido, la relación del auditor con su cliente es igual que la de un periodista respecto a sus fuentes, un médico respecto sus pacientes, o un sacerdote respecto sus feligreses.

3.5.2. IMPARCIALIDAD

Es habitual que el auditor reciba presiones tanto del cliente como de su propio equipo, así que deberá ser objetivo y no dejarse influenciar por esas presiones externas, vengan de donde vengan. Por ejemplo, un caso muy común, es que el cliente insista en que le quiten vulnerabilidades del informe final, para que no consten, o consten con un riesgo menor del que en realidad tienen. En otras ocasiones, el comercial que logra captar un cliente presiona al técnico auditor para que rebaje su estimación en tiempo, y con esto, poder vender un proyecto con el coste más bajo posible, o bien, insiste en que el técnico trabaje horas extra, que no se cobran al cliente, para poder entregar el informe en menos tiempo. El auditor debe saber que la responsabilidad de hacer bien un trabajo tan delicado es enteramente suya, y por ello, no debe ceder a ninguna de estas presiones, ni cambiar sus informes. La ley le protegerá siempre que documente la verdad: aquello que las evidencias que ha tomado pueden probar. En caso de mentir, no. Hay que tener en cuenta que, aunque en la auditoría de tipo web no suele ocurrir, falsear datos en otros tipos de auditoría, como por ejemplo la financiera, supone directamente cometer un delito. En conclusión, nadie puede obligar al auditor a mentir o falsear datos.

3.6. INFORME DE RESULTADOS: ELABORACIÓN Y ESTRUCTURA

El informe de resultados reúne todo el trabajo realizado durante la auditoría y entra dentro del tiempo del proyecto. En este apartado veremos cómo elaborar el informe correctamente, cómo presentar los resultados de manera comprensible para quién lo vaya a leer, y cuál es la estructura típica y los contenidos o secciones que debe presentar [8].

Hay muchos formatos de informe posibles. Cada empresa suele usar su propia plantilla y añadir o eliminar los componentes que considere más idóneos, pero en general, las secciones que se deben incluir para documentar un test de penetración web, son las siguientes:

3.6.1. RESUMEN EJECUTIVO

Se trata de un pequeño resumen, de una o dos páginas, que cuenta los hallazgos más importantes que se han encontrado. Está dirigido a los directivos o ejecutivos de la empresa cliente, que no necesitan leer todo el informe técnico, sino hacerse una idea global del riesgo al que se enfrentan, y entender el impacto en el negocio que supone cada resultado obtenido. Este apartado describe las conclusiones en un lenguaje sencillo, dirigido a personas sin conocimientos técnicos. Se incluyen gráficos y tablas que muestren el nivel de riesgo de cada objetivo, la fecha de realización, quienes han participado en el proyecto, y las acciones de contingencia que habría que tomar para mitigar el peligro o problema. Éstas se dan como actuaciones recomendadas.

3.6.2. CONSIDERACIONES TÉCNICAS GENERALES

Esta sección es una introducción breve dirigida a los responsables técnicos, que requieren conocer una serie de detalles que no se describen en el resumen ejecutivo, por

ejemplo, detalles del alcance del proyecto, los objetivos perseguidos, la disponibilidad del sistema o cuándo tuvo lugar el proyecto y su duración, y cualquier advertencia que se considere importante. También incluye un resumen de los hallazgos encontrados más importantes, pero explicados a nivel más técnico que en el caso anterior. Se puede explicar la metodología seguida por el equipo y listar los hallazgos según la fase en que fueron encontrados, o también aportar un inventario de sistemas auditados con detalles que identifiquen cada máquina, como su dirección IP, nombre, etc.

3.6.3. HALLAZGOS DURANTE LA EVALUACIÓN

Esta sección describe todos los detalles técnicos sobre los hallazgos encontrados, listados y ordenados por su nivel de riesgo: primero los altos, luego los medios, y por último los bajos. Cada hallazgo irá acompañado de: un identificador, para poder referenciarlos, los elementos afectados por él, un indicador del impacto que tienen sobre el sistema, y las recomendaciones necesarias para resolverlos, incluyendo fuentes donde se hayan publicado las soluciones conocidas, si es que las hay. Aquí es donde incluiríamos las evidencias recogidas, y los detalles técnicos descriptivos para que el equipo técnico del cliente sea capaz de reproducir la incidencia, entenderla, y resolverla.

3.6.4. HERRAMIENTAS UTILIZADAS

Esta sección es opcional. Describe las herramientas, comerciales y de código abierto, que han sido utilizadas para realizar la auditoría. Si se ha creado código a propósito para explotar la aplicación, podría incluirse aquí o bien en un apéndice o anexo.

3.6.5. CONCLUSIONES Y APÉNDICES

Esta sección es opcional. Se puede incluir para recoger un resumen final de conclusiones de no más de una página, y varios apéndices o anexos que amplíen la información dada.

CAPÍTULO 4. METODOLOGÍA. FASE DE RECONOCIMIENTO

La finalidad de esta primera fase, como ya dijimos, es recolectar toda la información que sea posible acerca del objetivo. En el mundo empresarial es habitual que el cliente proporcione al auditor toda la información que necesita relativa al alcance, ya que así ahorra en tiempo y, por tanto, en coste. Los auditores empezarían así su análisis directamente en la segunda fase. Sin embargo, estos atajos en la metodología no son recomendables, porque podemos pasar por alto algún detalle importante. Hay que pensar que un atacante potencial no tendrá ningún tipo de restricción temporal, por lo que será más fácil que él sí encuentre esos detalles.

Para explicar esta fase vamos a utilizar webs públicas en lugar de las de nuestro cliente, dado que las pruebas a realizar no entrañan ningún riesgo ni vulneran la confidencialidad del sitio. De este modo, evitamos ofuscar todos los hallazgos de esta fase por ser datos sensibles del cliente. Por ejemplo, vamos a utilizar la web oficial de la guía OWASP en la que estamos basando este método[11], y la web de la Wikipedia, cuyos datos se pueden ver publicados en su propia web[12].

El tipo de reconocimiento que vamos a practicar en nuestra auditoría será de tipo pasivo, donde el objetivo está prefijado de antemano y no hay que hacer una búsqueda indiscriminada por la red (la empresa nos ha proporcionado las tres URLs que conforman el alcance). No obstante, para mostrar la metodología, simularemos lo que sería el inicio de un ataque de tipo caja negra contra una web o contra una empresa.

El propósito de este ejercicio es identificar direcciones IP, nombres de máquinas, infraestructura de la red, perfiles y configuración de servidores, software, y cualquier otro dato involucrado con la aplicación web objetivo. Esto lo lograremos a través del uso de diferentes herramientas, la mayoría de ellas disponibles en todos los sistemas operativos. Vamos a describir algunas de ellas, de uso más frecuente en esta fase, y a mostrar un ejemplo práctico de su funcionalidad.

4.1. DEFINICIÓN DEL ALCANCE

Antes de comenzar a utilizar cualquier herramienta, deberemos definir el alcance del test de intrusión con el cliente, para que éste sepa con exactitud lo que se va a hacer, y lo que no se va a hacer [15]. En el alcance se decidirá si la prueba va a consistir en una caja negra, blanca o gris. Según la decisión que se tome, el auditor dispondrá de más información de entrada, o menos.

Respecto a la información facilitada, será el propietario o responsable de los sistemas o aplicaciones del cliente quien deberá aportar al auditor la dirección IP de los servidores o máquinas concretas que se deseen revisar, los rangos de direcciones de interés, si es que se trata de más de una máquina, el nombre del dominio o de la máquina a auditar, o bien la URL que se quiere poner a prueba. También es aconsejable aportar el código fuente de la aplicación, ya que podrían existir vulnerabilidades dentro del mismo que se podrían perder en un caso de prueba

[11] www.owasp.org
[12] www.es.wikipedia.org

de caja negra. Sin embargo, este tipo de revisión no se suele solicitar, porque revisar el código fuente lleva demasiado tiempo. El tiempo total de la auditoría aumentaría considerablemente y, con ello, también el coste. En lugar de esto, el cliente prefiere proporcionar documentación de la aplicación donde se detalle qué hace, cuáles son los puntos de entrada, etc.

En el caso de que estemos auditando una aplicación, puede ocurrir que la misma se encuentre alojada en un único servidor, o balanceada entre múltiples servidores. En estos casos es conveniente dar a conocer al auditor la tipología de red interna, cuáles son los equipos involucrados, y cómo conectan entre sí. Si no se aporta este conocimiento, el auditor tendrá que averiguarlo a partir de la información que se le suministre sobre la aplicación objetivo. Una vez obtenga la IP de las máquinas que alojan la aplicación, un buen profesional elegirá, entre ellas, la máquina más vulnerable como punto de entrada al sistema. En este caso, es conveniente que el auditor avise de sus descubrimientos antes de atacar una máquina que no figuraba en la definición de alcance inicial. Si el cliente da su visto bueno para que el ataque se produzca en la nueva máquina descubierta, deberemos incluir esta máquina en una versión actualizada del alcance. Debemos asegurarnos de que queda constancia por escrito de que hemos obtenido permiso expreso del cliente y que este es conocedor de la situación.

El alcance, además, también define los tipos de pruebas que deben hacerse y los tipos que quedarán excluidos. El auditor puede averiguar qué inquietudes tiene el cliente a través de una serie de preguntas acerca de lo que le preocupa. Es importante saber preguntar para que el alcance acordado no quede impreciso y poco claro para las partes. El auditor debe asegurarse, de manera explícita, de que aquellos elementos especialmente sensibles, que no deban ser auditados, queden fuera del alcance. Crear una lista de exclusiones es una buena idea. Crear un cuestionario tipo también será una gran ayuda para el auditor.

Si la web del cliente que nos contrata está alojada en un servidor perteneciente a otra entidad hay que avisar también al propietario de esta tercera entidad, ya que en su servidor pueden tener alojadas webs de otros clientes. Además, el auditor podría encontrar vulnerabilidades relacionadas con la configuración del servidor, y en ese caso, sería competencia de ésta tercera entidad, no de nuestro cliente, aplicar las correcciones y recomendaciones.

El sector de la empresa a auditar determinará en muchos casos el grupo de vulnerabilidades que el cliente prefiere evitar, y por tanto, el grupo que será prioritario en las pruebas. Por ejemplo, una empresa financiera, o dedicada a la investigación, estará muy preocupada por la divulgación de datos confidenciales; una empresa de venta de productos on-line, o una escuela que imparte cursos on-line, estará más preocupada por la interrupción del servicio o la caída de sus servidores.

En el presente ejercicio vamos a hacer una prueba de tipo caja negra donde sólo se nos ha proporcionado el nombre de la empresa, el nombre de los dos socios principales, y la URL del servidor que aloja las tres páginas web objetivo. Sabemos que el servidor pertenece también a la empresa, por lo que no existen terceras partes, y su negocio pertenece al sector de las nuevas tecnologías, por lo que será importante su presencia constante en la red y transmitir el mensaje de que sus productos son seguros y fiables.

4.2. BÚSQUEDAS EN REGISTROS DE INTERNET

En este punto, ya hemos acordado el alcance con el cliente y ya disponemos de los detalles que el cliente nos ha facilitado. A partir de ahora, nos disponemos a averiguar más

información sobre el objetivo, toda la que podamos recabar, con el uso de diferentes herramientas. Hay que identificar qué tipo de servidor es la máquina, bajo qué sistema operativo funciona, qué servicios tiene o qué puertos están abiertos, cuál es su configuración, con qué otras máquinas conecta, etc. Es aconsejable para el auditor verificar que la máquina objetivo sea realmente de su cliente. A veces, éste puede cometer errores al proporcionarnos la información. No verificar los datos puede llevar al auditor a realizar su trabajo sobre una máquina que no pertenece al cliente.

Podríamos agrupar los datos que encontremos en dos grupos: los que son públicos, y los que no lo son, o no deberían serlo.

4.2.1. WHOIS

La herramienta "Whois" [16] se utiliza para averiguar nombres de dominio, información de contacto, para identificar quienes son los propietarios de las direcciones IP de un dominio, o cuáles son los servidores DNS autoritativos (servidores primarios que almacenan los registros DNS). Esta información puede llevar al atacante a encontrar otros objetivos que atacar, como otras IPs del mismo propietario. Ejecutamos en nuestra máquina virtual "Kali", el siguiente comando:

```
# whois owasp.org
```

Obtenemos el resultado que se ve en la imagen, donde podemos identificar algunos datos de interés:

Figura 2: whois ejecutado sobre un terminal de Kali virtualizado sobre VmWarePlayer

En amarillo resaltamos los datos que más pueden interesarnos, y que podrían llevarnos a ampliar la búsqueda y a encontrar nuevos hallazgos.

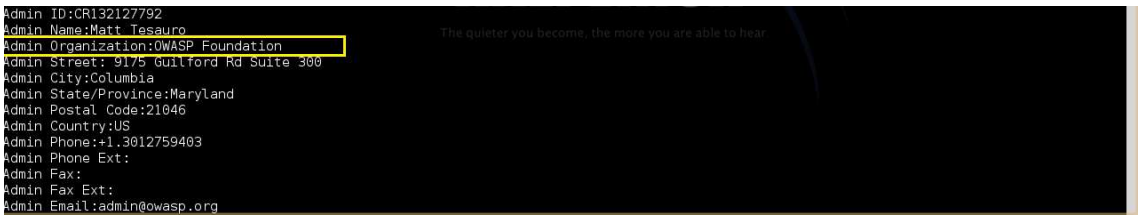


Figura 3: continuación del comando whois sobre la web oficial de Owasp

La siguiente imagen corresponde al mismo resultado, donde también resaltamos los nombres de los servidores de nombres (DNS) que tiene la organización Owasp:

Figura 4: whois sobre Owasp donde vemos sus dos nombres de DNS

Podemos ver otro ejemplo visitando la siguiente web de wikipedia: http://es.wikipedia.org/wiki/WHOIS. En ella se muestra el resultado del comando "whois" ejecutado sobre su propia web, en la sección "consulta de ejemplo".

4.2.2. NSLOOKUP

Esta herramienta se usa para hacer consultas DNS y obtener la dirección IP asociada a un nombre, o viceversa [17]. Igual que "whois", se puede ejecutar por línea de comandos y también como servicio basado en web. Un atacante preferiría el tipo basado en web para enmascarar su propia IP y no ser detectado. En línea de comandos escribimos:

```
# nslookup owasp.org
```

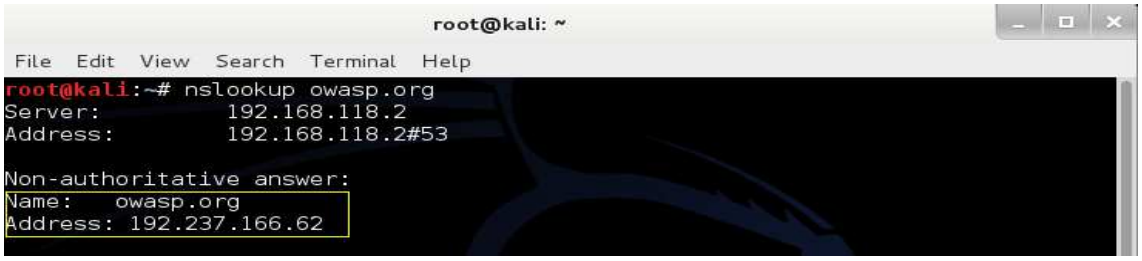


Figura 5: respuesta no autoritativa al comando nslookup

En la imagen, recuadrado en amarillo, vemos la dirección IP del sitio "owasp.org". Si en lugar de lanzar el comando por Linux lo hacemos por web [18], por ejemplo desde www.networking.ringofsaturn.com/Tools/nslookup.php, (hay muchas otras páginas que ofrecen este tipo de servicios), obtenemos la misma IP para Owasp, pero vemos que la IP del servidor que lanza el comando cambia de la puerta de enlace de nuestro equipo [192.168.118.2] a una dirección del dominio que aloja la web "ringofsaturn" [71.252.219.43]:

Figura 6: rellenamos "hostname/domain" y " tipo de consulta". El server viene por defecto.

Query Results

```
Server:         tethys.ringofsaturn.com
Address:        71.252.219.43#53

Non-authoritative answer:
Name:   owasp.org
Address: 192.237.166.62
```

Figura 7: resultado tras clicar el botón de "submit"

En ambos casos, por terminal o por web, tenemos opciones para cambiar el tipo de consulta que queremos ver: A, NS, PTR, SOA, MX, etc. Según la opción elegida nos aparecerán datos distintos. Por ejemplo, la siguiente imagen nos muestra el resultado de la consulta tipo NS, donde se solicita ver cuáles son los nombres de los servidores de DNS de Owasp:

```
Server:         tethys.ringofsaturn.com
Address:        71.252.219.43#53

Non-authoritative answer:
owasp.org       nameserver = dns1.stabletransit.com.
owasp.org       nameserver = dns2.stabletransit.com.

Authoritative answers can be found from:
dns1.stabletransit.com  internet address = 69.20.95.4
dns2.stabletransit.com  internet address = 65.61.188.4
```

Figura 8: nombres de dos servidores DNS y sus respectivas IPs

Como auditores, nos interesan tantos datos como podamos averiguar, exactamente igual que haría un atacante malintencionado. Por ello, elegiríamos la opción ANY para que nos muestre toda la información que exista sobre el servidor objetivo. En la imagen podemos ver, resaltados en rojo, los hallazgos encontrados: dirección IP pública, nombres de servidores de correo (por el nombre de dominio descubrimos que no tienen un servidor propio sino que han contratado el servicio de correo a Google), nombres de los dos servidores de DNS y la respuesta autoritativa. El resultado de la consulta aparece en la siguiente imagen:

```
Server:         tethys.ringofsaturn.com
Address:        71.252.219.43#53

Non-authoritative answer:
owasp.org       text = "google-site-verification=I9qx_X9EKlR_rfceG25-iXHBXJvLrmeNbkEdyl82iI"
owasp.org       text = "v=spf1 include:aspmx.googlemail.com ~all"
Name:   owasp.org
Address: 192.237.166.62
owasp.org       has AAAA address 2001:4801:7821:77:cd2c:d9de:ff10:170e
owasp.org       mail exchanger = 20 ALT1.ASPMX.L.GOOGLE.com.
owasp.org       mail exchanger = 20 ALT2.ASPMX.L.GOOGLE.com.
owasp.org       mail exchanger = 30 ASPMX2.GOOGLEMAIL.com.
owasp.org       mail exchanger = 30 ASPMX4.GOOGLEMAIL.com.
owasp.org       mail exchanger = 30 ASPMX5.GOOGLEMAIL.com.
owasp.org       mail exchanger = 10 ASPMX.L.GOOGLE.com.
owasp.org       mail exchanger = 30 ASPMX3.GOOGLEMAIL.com.
owasp.org
        origin = dns1.stabletransit.com
        mail addr = ipadmin.stabletransit.com
        serial = 1410656493
        refresh = 3600
        retry = 300
        expire = 1814400
        minimum = 300
owasp.org       nameserver = dns2.stabletransit.com.
owasp.org       nameserver = dns1.stabletransit.com.

Authoritative answers can be found from:
owasp.org       nameserver = dns2.stabletransit.com.
owasp.org       nameserver = dns1.stabletransit.com.
```

Figura 9: consulta ANY para el comando nslookup

Como segundo ejemplo podemos ver el resultado del comando sobre la web de wikipedia, publicado por ellos mismos: http://es.wikipedia.org/wiki/Nslookup.

4.2.3. DIG

Dig[13] (Domain Information Groper) es una herramienta para Unix/Linux que reemplaza al "nslookup". Es más moderna pero sirve para lo mismo: realizar consultas DNS. En Windows también podemos encontrarla pero no por línea de comandos, sino a través de las distintas webs de herramientas[14]. Siguiendo con nuestro ejemplo, mostramos a continuación el resultado de aplicar esta herramienta a la web de Owasp desde el sistema Linux virtual:

```
# dig ANY owasp.org
```

Obtenemos el siguiente resultado, en el que vemos que los datos coinciden con los encontrados en el ejercicio anterior:

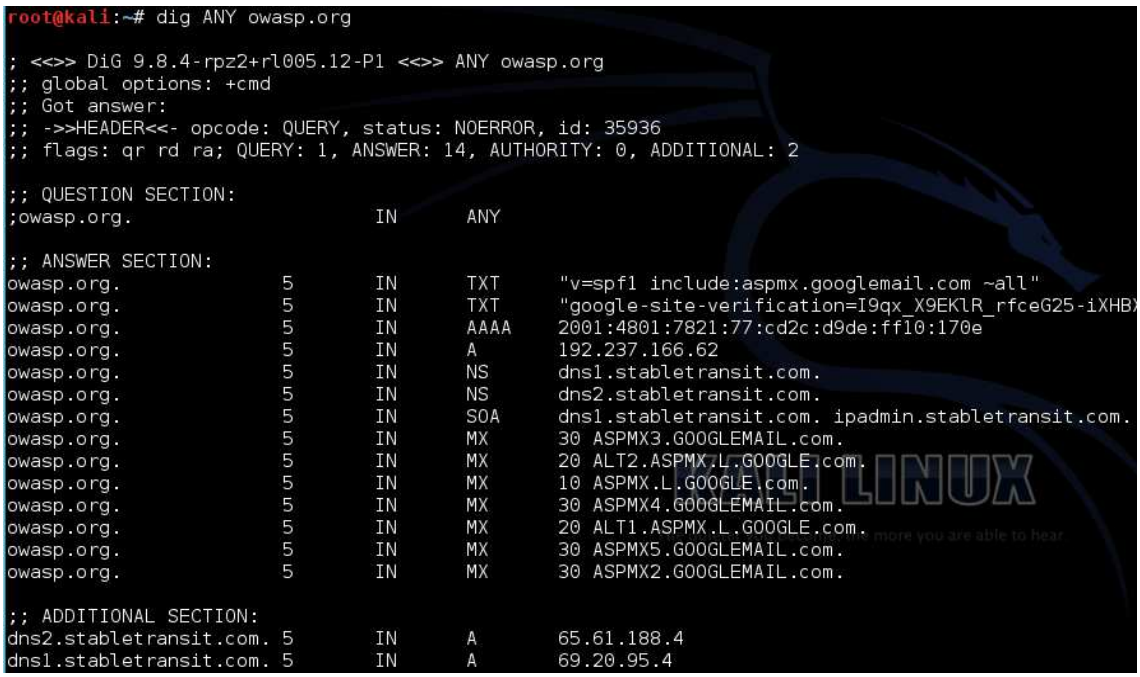


Figura 10: ejecución del comando "dig"

Existen otras herramientas propias de los sistemas operativos además de las que hemos mostrado (host, traceroute, etc), pero para web, éstas son las que nos dan mejores resultados.

4.3. CONSULTAS EN PÁGINAS PÚBLICAS

Con frecuencia podemos encontrar en la red mucha información pública asociada a la entidad que estamos auditando. Esa información nos conducirá a nuevos hallazgos sin necesidad de alertar al objetivo de que está siendo objeto de una actividad maliciosa. Algunos ejemplos de sitios web dónde podemos buscar son:

[13] https://siliconhosting.com/kb/questions/360/Comando+dig
[14] http://networking.ringofsaturn.com/Tools/dig.php

4.3.1. PÁGINAS DE NOTICIAS

¿Aparece nuestro objetivo en la prensa? ¿Y por qué razón? ¿Quizá algo que podamos aprovechar? Es habitual que una gran firma (Apple, Telefónica, Microsoft, etc) haga un comunicado o dé una rueda de prensa para anunciar su nueva gama de productos, el lanzamiento de la última versión de su producto estrella, la absorción de otra compañía del sector, o la puesta en marcha de una nueva tecnología y, por tanto, de un nuevo dispositivo. Estos anuncios pueden revelar mucha información útil, como por ejemplo, con qué tecnologías, lenguajes, etc, están trabajando.

4.3.2. GRUPOS DE SOPORTE TÉCNICO

En estos grupos los usuarios suelen revelar qué problemas han encontrado en su nueva adquisición, y esto puede indicar alguna vulnerabilidad: un fallo de funcionamiento, un comportamiento inesperado, etc.

4.3.3. LISTAS DE CORREOS DE TEMAS TÉCNICOS

En estas listas, de forma muy parecida al caso anterior, alguien puede revelar un error encontrado en un dispositivo o aplicación antes de que haya sido reparado por el propietario. Incluso hay empresas que lanzan al mercado su producto sin que éste esté probado adecuadamente, y usan estas listas para que sean sus propios clientes, tras encontrar los errores al usar el producto, quienes los reporten a la empresa. Así, la empresa va depurando su producto en base a los hallazgos de sus propios clientes, y sacando nuevas versiones que arreglan los fallos de las versiones anteriores. En ocasiones son los propios clientes quienes crean un parche corrector de la aplicación, y lo publican abiertamente. Así que, en estas listas se puede encontrar gran cantidad de información como fragmentos de código, ficheros de configuración, datos sobre la arquitectura de una aplicación web, etc. Esto es bastante común en productos con licencia de software libre.

4.3.4. REDES SOCIALES PROFESIONALES TIPO LINKED-IN

En éstas, los profesionales indican a una red de socios sus habilidades, conocimientos, y para qué empresa trabajan; podemos ver quiénes trabajan para nuestra empresa objetivo y en qué campos son expertos. En conclusión, podemos suponer que la empresa podría ser más floja en aquellos campos que sus profesionales en plantilla no dominan. Por ejemplo, si una empresa de software cuenta con expertos en lenguaje Java, pero no cuenta con ninguno en otros lenguajes de programación, como PHP o .NET, lo primero que podemos deducir es cuál es el lenguaje o la tecnología con la que trabajan y crean sus productos. Lo siguiente que podríamos decir es en qué campos podrían ser más vulnerables que en otros.

4.3.5. PÁGINAS DE OFERTAS DE EMPLEO

En estos tipos de páginas podemos encontrar información actualizada, y muy específica, acerca de la empresa objetivo. En estas páginas, las empresas publican perfiles muy completos de lo que buscan, perfiles donde suelen revelar, sin darse cuenta, muchos datos aprovechables por un atacante.

La empresa indica con gran nivel de detalle las tecnologías que utiliza o desarrolla, las versiones de los programas y servidores que mantiene, o si sus preferencias son de un determinado sistema operativo u otro (pro Linux, pro Windows…). Gracias al listado de

requerimientos podemos ver para qué áreas necesita profesionales, y gracias a la descripción de su actividad podemos deducir en qué áreas puede estar, o no estar, bien cubierta.

También revelan el perfil de trabajador que compone su equipo técnico, o el que desearían tener: si quieren gente joven y dinámica, si cuentan con mucha o poca experiencia, si se renueva a la plantilla con cursos de formación o no, etc. En las ofertas de cargos responsables podemos hacernos una idea de lo grande que pueden ser sus departamentos o grupos de trabajo. Por ejemplo, si se busca a un analista, la oferta podría indicar a cuánta gente tendría a su cargo. Si nos fijamos en las fechas, podemos averiguar la duración de los proyectos que la empresa pretende emprender y si son proyectos nuevos o no.

A veces, los proveedores o consorcios asociados a la empresa pueden ser más vulnerables que la propia empresa. Conocerlos podría constituir un punto de entrada fácil para, desde allí, atacar a la empresa más fuerte.

4.3.6. MOTORES DE BÚSQUEDA GENERALES

Esta es una de las opciones más fructíferas a la hora de rastrear información pública por la red, por lo que la trataremos más a fondo en el siguiente apartado.

4.4. BÚSQUEDAS CON MOTORES DE BÚSQUEDA

Los motores de búsqueda en concreto son una gran fuente de datos, eficaces debido a la cantidad de información que la gente revela, a veces con un propósito, y otras veces sin ser conscientes. En esta sección vamos a centrarnos en cómo buscar información con Google.

4.4.1. ROBOTS.TXT

Robots.txt es un fichero de texto que sirve para darle instrucciones a los robots de los indexadores de Google, Yahoo, y otros buscadores [20]. Les indican qué contenidos deben, o no deben, rastrear. El texto se compone de una lista de directivas o comandos que realizan, cada uno, una función determinada.

Los comandos son: user-agent, allow, y disallow.

- **User-agent:** indica a qué robot se le va a aplicar la regla escrita a continuación de los dos puntos. A menudo vemos un asterisco en esta posición "*". Es un carácter comodín que sirve para indicar que la regla se aplicará sobre cualquier robot.

- **Allow:** indica a los robots que está permitido indexar el contenido que viene a continuación de los dos puntos.

- **Disallow:** nos indica lo contrario de "Allow", que no está permitido indexar el contenido que viene a continuación de los dos puntos. Si vemos el comodín "/" quiere decir que se niega el acceso, a todos los robots, en todos los archivos del directorio raíz. Si aparece un nombre tras la barra, el acceso se negaría sólo en ese directorio concreto.

Si editamos por ejemplo, el fichero robots.txt de la página web de Wikipedia (www.es.wikipedia.org/robots.txt), nos aparece un texto muy largo con muchas secciones y directivas:

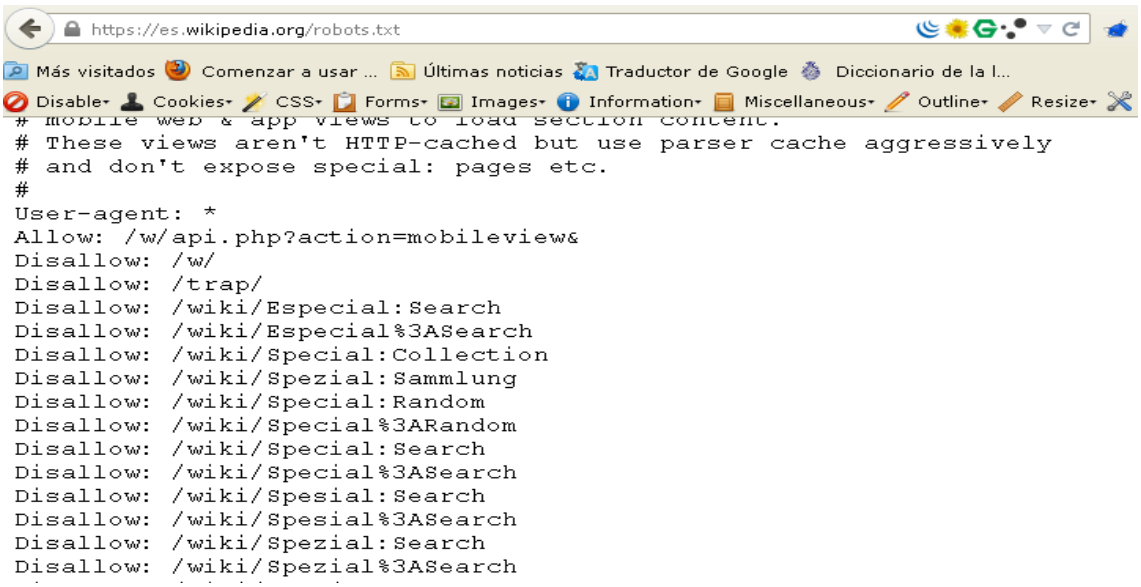


Figura 11: sección ejemplo del fichero robots.txt de la página de wikipedia

Los recursos que aparecen a continuación de las directivas "Disallow:" ayudan al atacante (y al auditor) a conocer contenidos de la web de antemano. El hacker se preguntará por qué hay directorios que han sido deshabilitados, así que el auditor deberá preguntárselo también. Por esta razón, intentaremos acceder a esos contenidos ocultos desde el buscador, y averiguar la información que guardan. Para esto se pueden utilizar herramientas de fuerza bruta sobre esos directorios que, aunque no están indexados por el buscador, sí que existen y pueden ser accedidos. Dos herramientas muy comunes para este tipo de análisis son Dirbuster[15] (ahora integrado en ZAP de Owasp), y Dirb[16]. Sobre este tipo de ataques y sobre las herramientas empleadas se hablará en detalle más adelante.

4.4.2. DIRECTIVAS Y OPERADORES

Google tiene una serie de directivas y operadores para facilitar las búsquedas y para concretarlas. Cada buscador tiene las suyas propias pero algunas de ellas coinciden con las de otros buscadores. Estos mecanismos sirven para limitar los resultados de las búsquedas. Con ello, permiten al usuario centrarse en un objetivo preciso y le evitan hacer a mano una criba entre un montón de resultados hallados indiscriminadamente. Para un auditor, las directivas y operadores serán de gran utilidad durante la fase de reconocimiento, ya que le permiten focalizarse en su objetivo y ahorrar mucho tiempo. Algunas de las directivas y operadores más usuales se explican en el anexo 5, donde se muestran algunos ejemplos [21, 24].

La ventaja de esta forma de recolectar datos es que puede llegar a reunirse información muy útil sin conectar ni una sola vez con la web o IP objetivo. La segunda ventaja es que los motores de búsqueda son muy potentes e indexan gran cantidad de información, incluso la que no es visible en una página, por lo que es posible llegar a averiguar mucha información partiendo de entrada con muy pocos datos. En concreto, Google tiene tantas posibilidades como herramienta de hacking que podríamos dedicar el proyecto completo sólo a él.

[15] http://sectools.org/tool/dirbuster/
[16] http://dirb.sourceforge.net/about.html

4.5. BÚSQUEDA DE SUB-DOMINIOS

Una forma de encontrar nuevas máquinas objetivo es comprobar si el dominio que estamos auditando tiene sub-dominios asociados. Los sub-dominios pertenecerán al mismo propietario del dominio raíz. Un ejemplo de sub-dominio de "owasp.org" sería "www.owasp.org", o también "mail.owasp.org". Para encontrar estos sub-dominios podemos utilizar la herramienta de escaneo "Fierce Domain"[17].

4.5.1. FIERCE DOMAIN SCANNER

Se trata de una herramienta que hace un escaneo para encontrar servidores hosts dentro de un dominio o de una red [25]. Usa una lista de nombres comunes de servidores, o de prefijos habituales (como mail, ns, smtp), para ver si existen y tienen IP válida. Cuando encuentra IPs válidas, realiza una búsqueda sobre varios números IP previos al encontrado y también consecutivos. Por ejemplo, si encuentra la IP 84.10.20.30, buscará en el rango desde la 84.10.20.25, hasta la 84.10.20.35. Además, ésta herramienta está escrita en lenguaje "perl", así que es soportada por muchas plataformas diferentes.

Esta herramienta se suele emplear sobre todo por los atacantes malintencionados y durante las auditorías profesionales de caja negra. En este tipo de auditorías el cliente puede proporcionarnos sólo el nombre de la empresa o un nombre de dominio DNS. A continuación, vemos el resultado de aplicar la herramienta al dominio de Owasp. Lanzamos el siguiente comando contra el dominio raíz:

```
# fierce –dns owasp.org
```

Figura 12: ejemplo de uso de la herramienta "fierce"

[17] Disponible en la web http://ha.ckers.org/fierce/

Obtenemos mucha información nueva. Ya conocíamos los dos servidores de DNS de Owasp, pero ahora además hemos encontrado una serie de sub-dominios, ocho en total, con sus respectivas direcciones IP, que pertenecen a cinco sub-redes distintas. La herramienta nos agrupa estas sub-redes, nos muestra sus rangos enteros, y nos sugiere qué otras herramientas o comandos usar sobre ellas, como nmap o unicornscan. También podemos usar el comando con otras opciones como por ejemplo, lanzar un segundo escaneo sobre los rangos de IPs que ha encontrado previamente:

```
# fierce –range 192.237.166.0-255 –dnsserver 192.237.166.62
```

4.6 ELABORACIÓN DE DICCIONARIOS

La elaboración de diccionarios es otra técnica de hacking que puede resultar útil. Consiste en crear un diccionario, o listado, con todas las palabras que aparecen en la web objetivo, para después aplicar sobre ese diccionario técnicas de ataque por fuerza bruta [26].

4.6.1 CEWL - CUSTOM WORD LIST GENERATOR

La herramienta CeWL[18] es uno de los generadores de listas de palabras. Dada una URL objetivo, la analiza recorriendo todos sus enlaces externos, y devuelve un listado de palabras encontradas, que después será utilizado por otras herramientas para la preparación de ataques de fuerza bruta, bien de directorios o bien de contraseñas. El modo de empleo es el habitual con otros comandos: `cewl -[opciones] URL`. Como ejemplo de su uso mostramos el siguiente comando:

```
# cewl –w fichero.txt dominio_objetivo.com
```

La opción `[-w fichero]`, escribe en un fichero de texto `[fichero.txt]` todas las palabras que el programa haya podido encontrar en la web indicada como objetivo: `[dominio_objetivo.com]`. Este tipo de rastreos, hoy día, vienen incluidos en herramientas para mapeo como puede ser ZAP, cuyo uso se explicará más adelante.

[18] http://digi.ninja/projects/cewl.php

CAPÍTULO 5. METODOLOGÍA. FASE DE MAPEADO

En esta segunda fase, lo principal es familiarizarnos con la aplicación web objetivo y el servidor web que la aloja. Para ello se emplearán distintas herramientas que nos permitirán conocer mejor la tecnología utilizada por la aplicación. De la misma forma, se navegará por la web objetivo como si fuéramos un usuario cualquiera, pero utilizando un proxy web que nos permita conocer y modificar las interacciones entre nuestro navegador y la aplicación web objetivo. Al final de esta fase, habremos recorrido toda la web al completo y tendremos un mapa que nos mostrará el flujo lógico entre las distintas páginas, y las relaciones entre éstas. También habremos identificado todas las máquinas que entran en juego con la aplicación.

El resultado de las tareas de esta fase permitirá al auditor conocer con profundidad la plataforma y la aplicación, con el objetivo de encontrar vulnerabilidades en la siguiente fase de descubrimiento. Como recomendación, es una buena práctica que el auditor vaya anotando aquellos puntos de la web donde pueda intuir que la probabilidad de encontrar una vulnerabilidad es alta. Hay webs cuyo recorrido completo puede llegar a ser muy amplio, e incluso ocupar un buen número de horas de dedicación. Por esto, es de gran ayuda marcar aquellas secciones o enlaces que creamos que puedan ser delicadas. Por ejemplo, si viajamos por un menú que nos lleva a una página estática, que sólo contiene texto, es probable que en esta sección no encontremos nada vulnerable, ya que no hay ninguna aplicación detrás, ni ningún recurso dinámico que pueda generar un fallo de uso. Sin embargo, si nos encontramos un formulario que nos permite introducir texto, o una opción que nos permite cargar o descargar ficheros, la probabilidad de encontrar en esta página una o más vulnerabilidades es más alta, y tendremos que anotarla para volver sobre ella más tarde.

Recordemos que en esta fase todavía no vamos a poner en práctica los ataques. Vamos a limitarnos a recorrer toda la aplicación y a conocerla a fondo, sin entrar a probar sobre ella cada posibilidad de fallo que se nos ocurra. Esto se hará más tarde, en las fases siguientes. Lo que es más importante en esta fase es crearnos un mapa mental del sitio objetivo y separar las zonas con baja probabilidad de ser vulnerables, de las zonas con alta probabilidad de serlo. El auditor debe dar prioridad a estas últimas, y centrar sobre ellas su trabajo, ya que su tiempo es muy limitado. Saber gestionarlo bien le ayudará a realizar auditorías de mayor calidad.

El punto de partida del mapeo es el final del reconocimiento que debíamos hacer con anterioridad. En nuestro caso, durante el ejercicio previo hemos podido averiguar de nuestro cliente los siguientes datos: su dirección IP, su proveedor de internet, a nombre de quién está registrado el servidor principal, el nombre del dominio, el nombre y el número de servidores de DNS que posee, si cuenta con sub-dominios y cuáles son estos, y además, el diccionario de palabras que se utilizan en las páginas que estamos auditando. Además hemos comprobado que no tiene ninguna ruta deshabilitada en su fichero robots.txt y que tampoco entran en juego terceras entidades, ya que tanto las tres webs, como el servidor que las aloja, pertenecen al mismo propietario.

Hasta el momento, con la intención de resguardar la confidencialidad de los datos del cliente, hemos mantenido ocultos sus resultados, y hemos utilizado dos sitios web públicos para mostrar en imágenes las evidencias y los resultados que se obtienen al aplicarles las diferentes herramientas en la fase de reconocimiento. A partir de ahora empezaremos a mostrar los resultados del mapeo del sitio web del cliente, pero ofuscando todos los datos sensibles. Así pues, para no desvelar las verdaderas URLs auditadas, nos referiremos a ellas con los siguientes

nombres ficticios: fusionmetales.com para la primera web, escuelatest.com para la segunda web, y auditandowebs.com para la tercera web.

5.1. ESCANEO DE PUERTOS Y VERSIONES

Del conjunto de tareas que componen el mapeado, el escaneo de puertos sería la primera de ellas, aunque según la experiencia del auditor y el tipo de herramientas que use (dedicadas a una sola función o multifuncional), esta tarea podría solaparse con otras, o podríamos tener que repetirla en un momento más avanzado si, por ejemplo, descubrimos nuevas máquinas.

Cada máquina tiene un máximo de 65.535 puertos TCP, y otros tantos UDP, desde los que se comunica con las otras máquinas de la red. Se utilizan para realizar conexiones salientes, como cuando accedemos a un servidor, o para tener servicios a la escucha esperando conexiones entrantes. En cada máquina, lo óptimo y más seguro, es que los puertos que no se usen permanezcan cerrados. El escaneo de puertos lo que hace es comprobar qué puertos están abiertos y qué aplicaciones están escuchando por ellos. Si un atacante encuentra un servicio vulnerable detrás de un puerto abierto, su siguiente paso será encontrar la manera de explotarlo. De ahí que el escaneo de los puertos de un servidor sea una de las tareas del auditor web. El consejo es que cada servidor tenga abiertos sólo aquellos puertos que requiera para desempeñar su función. Por ejemplo, si nuestro servidor es web, necesita abrir el puerto 80, pero no necesita mantener abierto el puerto 21 con un servicio de FTP.

5.1.1. NMAP

Como herramienta para realizar escaneos de puertos, Nmap[19] es la más conocida, muy utilizada por los administradores de redes y sistemas. Su función principal es mostrar un listado de los puertos abiertos que tiene un sistema, y qué servicios están escuchando en esos puertos. Tiene múltiples opciones que nos permiten dar con información adicional, como puede ser el sistema operativo que está instalado en la máquina y su versión actual [27].

5.1.1.1. TCP SYN SCAN

Para un atacante, el comando Nmap tiene una gran desventaja: que puede ser detectado con facilidad por un IDS. Las empresas que quieren proteger sus servidores instalan en ellos estos IDS, que son sistemas de detección de intrusiones. Lo que hace Nmap es enviar un paquete a cada puerto de la máquina y esperar el paquete de respuesta. Esto genera un aumento significativo en el tráfico de red de ese servidor, lo que es detectado por el IDS. El IDS genera entonces una alerta, y el administrador de sistemas que monitoriza las alertas puede capturar la IP del atacante. Para evitar esto, y que el escaneo sea más difícil de detectar, se utiliza el escaneo de tipo TCP SYN scan, que no completa el establecimiento de la sesión TCP. En nuestro ejercicio, lanzamos el comando siguiente contra el servidor del cliente, al que vamos a llamar "dominio_cliente.com":

```
# nmap -sV -sS -O -sC --top-ports 4000 dominio_cliente.com -oA nmap-TCP4000
```

Incluimos en la orden las siguientes opciones:

-sV: pedimos a Nmap que trate de identificar las versiones de software empleado.

[19] http://nmap.org/

-sS: indica a Nmap que haga un escaneo del tipo TCP SYN scan. Envía un paquete TCP con el bit de SYN activado. Si el servidor responde con un SYN/ACK es que está habilitado y si no responde, o responde con un RESET, entonces está cerrado.

-O: Nmap identifica el sistema operativo de la máquina escaneada.

-sC: indica a Nmap que ejecute todos los scripts asociados (NSE scripts) al servicio encontrado.

--top-ports 4000[20]**:** indica a Nmap que escanee los 4.000 puertos más comunes. Según estudios estadísticos, un escaneo de puertos TCP de los 1.000 puertos más comunes ofrece el 93% de los puertos TCP abiertos. Nosotros hemos optado por ampliarlo a 4.000, para mejorar los resultados. En auditorías profesionales se suele emplear esta opción, porque las ventanas de tiempo son reducidas y un escaneo completo de 65.535 puertos es muy lento y apenas mejora el resultado final.

-oA file: sirve para copiar en un archivo llamado "file" el resultado del comando. Genera tres ficheros: un xml, y dos de texto, uno de ellos preparado para hacerle filtrados con el comando "grep[21]".

Figura 13: resultado del escaneo de puertos tipo TCP con nmap

El dominio del cliente aparece ofuscado para que no se pueda leer, así como su IP del servidor. Remarcado en amarillo vemos los datos que hemos logrado averiguar: una pequeña lista con el número de puerto, el estado en que están (si abiertos o cerrados), el servicio que escucha en ese puerto, y la versión del servicio. Más abajo vemos el sistema operativo que está usando la máquina, un Linux Debian a juzgar por las versiones de los servicios. Nos anotamos como datos de interés estos servicios y sus versiones. En la fase de descubrimiento deberemos averiguar si estos servicios, o las versiones instaladas, son vulnerables.

5.1.1.2. UDP SCAN

En el punto anterior, hemos pedido a Nmap un recorrido por los puertos de tipo TCP. Ahora vamos a lanzar otro escaneo similar, pero esta vez recorriendo los puertos de tipo UDP. Para ello, usamos la opción de Nmap **–sU**. En lugar de 4.000, bajamos a los 1.000 puertos más comunes. El resto de opciones las dejamos igual:

```
# nmap -sU -sV -O -sC --top-ports 1000 dominio_cliente.com -oA nmap-UDP1000
```

Obtenemos el siguiente resultado:

[20] http://nmap.org/5/
[21] http://linux.about.com/od/commands/l/blcmdl1_grep.htm

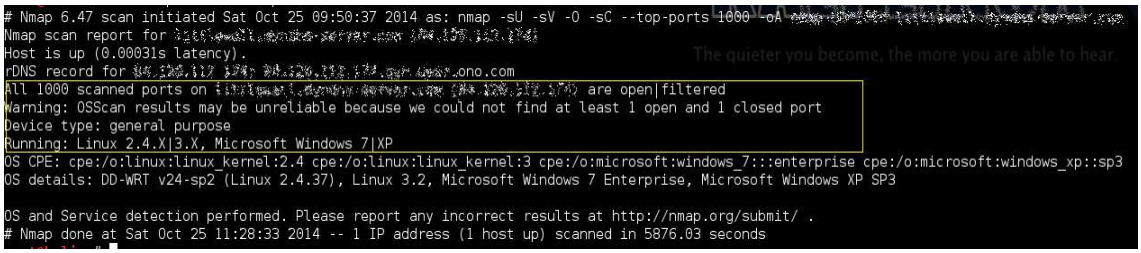


Figura 14: resultado del escaneo de puertos tipo UDP con nmap

El resultado no nos ha aportado nada nuevo. Parece que el servidor de nuestro cliente tiene todos los puertos UDP cerrados, aunque veamos en la primera línea recuadrada que están todos abiertos. Si fuera así, nos aparecería el puerto junto al servicio y la versión en una lista de puertos, por tanto, esta indicación no es correcta y deducimos que no hay servicios en uso por puertos UDP.

Además del escaneo de puertos, en este ejercicio hemos realizado simultáneamente el escaneo de versiones (tanto del sistema operativo, como de las aplicaciones y servicios instalados) que teníamos programado para el siguiente punto. Las opciones para Nmap nos han permitido realizar ambas tareas a la vez lanzando un único comando. Como ya disponemos de esta información no necesitamos aplicar otras herramientas para conseguirla, por lo que podemos pasar directamente a la siguiente tarea. Si no fuera así, tendríamos que buscar un poco más a fondo, pues es habitual que las versiones antiguas de una aplicación sean vulnerables donde las versiones actualizadas no lo son. En parte, se actualizan para corregirlas. Además, si una versión es antigua, lo más probable es que sus vulnerabilidades se hayan hecho públicas y sean conocidas, e incluso que se hayan publicado "exploits"[22].

Existen páginas web que recopilan este tipo de herramientas y las versiones del software al que afectan[23].

5.2. ANÁLISIS SSL

SSL (Secure Socket Layer) es un protocolo de cifrado que garantiza la confidencialidad de las comunicaciones por internet. Se utiliza para establecer una conexión segura entre cliente y servidor web por medio del protocolo HTTP. El objetivo del análisis SSL de esta sección es el de verificar la seguridad del soporte de SSL ofrecido por el servidor Web del cliente. Es conocido el hecho de que las primeras versiones de este protocolo eran fáciles de vulnerar, e incluso las versiones posteriores que usaban cifrados débiles. De ahí que el protocolo haya tenido que evolucionar rápidamente, para solventar todos estos problemas. En concreto se verificarán los siguientes aspectos:

- ¿El servidor soporta SSL? Es recomendable que todas las páginas web que disponen de información sensible soporten SSL, especialmente aquellas que tienen parte autenticada.
- ¿Las versiones de SSL soportadas son seguras? Las instalaciones modernas soportan SSLv3 y TLSv1. Cualquier servidor que soporte versiones anteriores como SSLv2 es inseguro y tiene vulnerabilidades conocidas.

[22] http://es.wikipedia.org/wiki/Exploit
[23] http://www.exploit-db.com/

- ¿Cuál es el tamaño de las claves y los tipos de cifrado soportados por el servidor? Algunos tipos de cifrado como el "NULL cipher" o el "RC4" son considerados inseguros y al respecto de las claves, también se considera inseguras las menores de 128 bits.
- ¿El certificado del servidor es aceptado por todos los navegadores o muestra algún tipo de error? Los certificados mostrados por los servidores confiables no deben mostrar ningún error que pueda hacer desconfiar al usuario.

5.2.1. TLSSLED

La herramienta TLSSled es un script para Linux, basado en Openssl y Sslscan, que sirve para analizar la seguridad de la implementación SSL/TLS de un servidor web objetivo [28]. El programa realiza una serie de comprobaciones para verificar todos los aspectos que hemos enumerado en el punto anterior. Verifica si el servidor objetivo soporta el protocolo SSL 2.0 o superior, si soporta el cifrado nulo (Null cipher) o algún otro tipo de los considerados débiles, si soporta tamaños de claves menores de 128 bits, o si el certificado digital está firmado con MD5, entre otras cosas. Existen otras herramientas similares para Windows, como veremos después.

Seguidamente, lanzamos esta herramienta contra el servidor de nuestro cliente y analizamos los resultados. En el comando, indicamos la ruta del servidor y el puerto que se corresponde al servicio SSL, generalmente el 443 de tipo UDP, o el 80 de tipo TCP:

```
# tlssled dominio_cliente.com 80
```

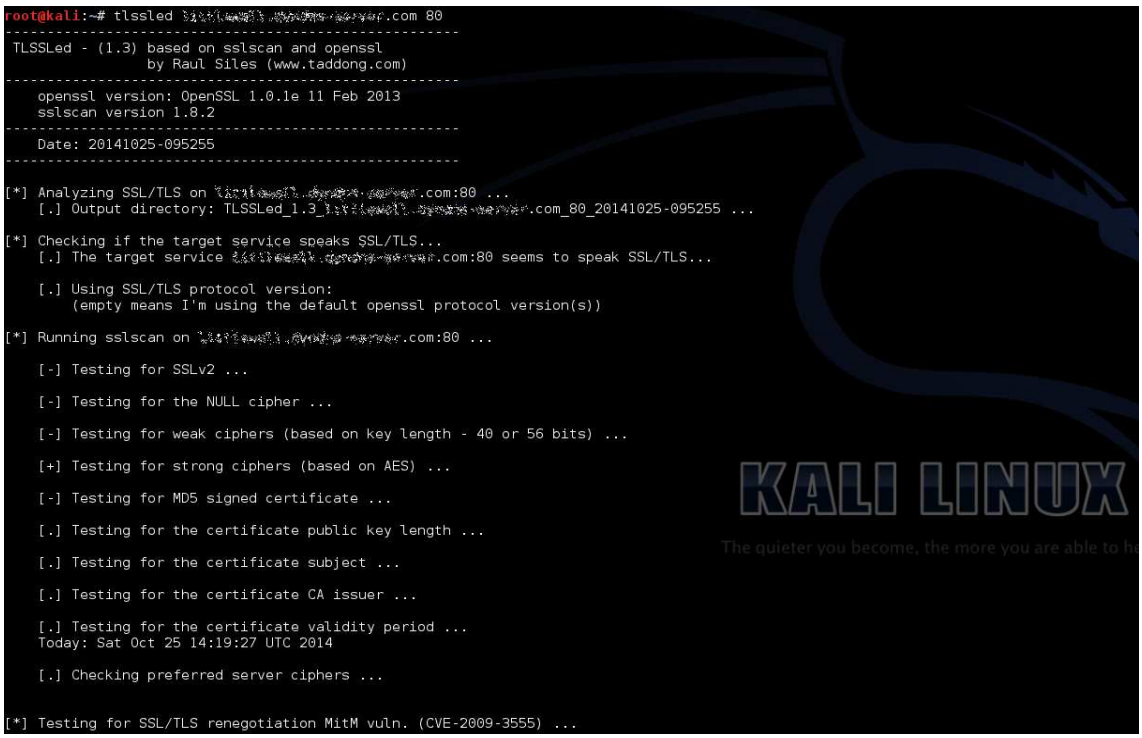


Figura 20: análisis SSL al puerto TCP 80 del cliente

En este caso se realizan muchas pruebas pero ninguna de ellas da ningún resultado. Concluimos que la comunicación TCP no soporta ningún tipo de SSL. Ahora comprobamos el resultado del puerto 443 UDP:

```
# tlssled dominio_cliente.com 443
```

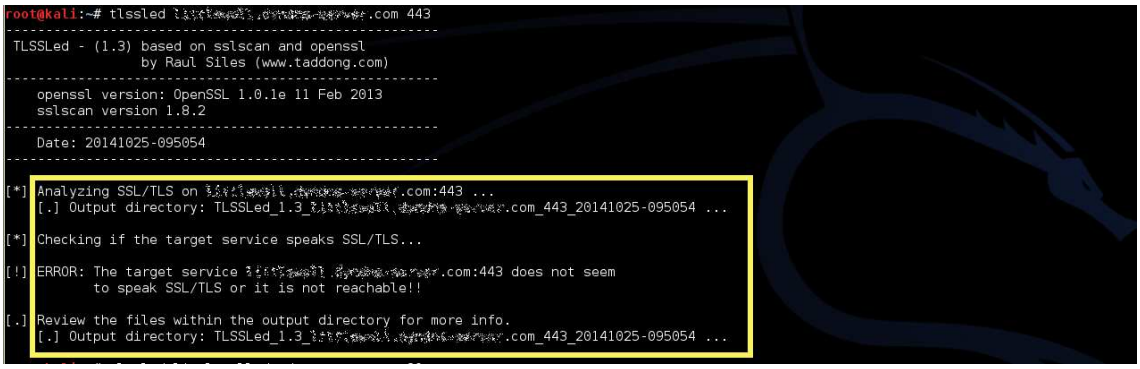


Figura 21: análisis SSL al puerto 443 del cliente

En la imagen obtenida para el puerto 443 vemos que el test finaliza sin dar ningún resultado. De esto interpretamos que el cliente no soporta ninguna de las versiones de SSL/TLS, ni las vulnerables ni las actuales. Esto quiere decir que la información que envía en una comunicación no va cifrada, por lo tanto, el servidor es vulnerable o podemos sospechar que estamos frente a un fallo en la seguridad.

Para que podamos apreciar la diferencia con el resultado de un servidor que sí soporta estos protocolos, vamos a mostrar el resultado de aplicar la herramienta a un servidor de Google cuya implementación sí es segura[24]:

```
# tlssled mail.google.com 443
```

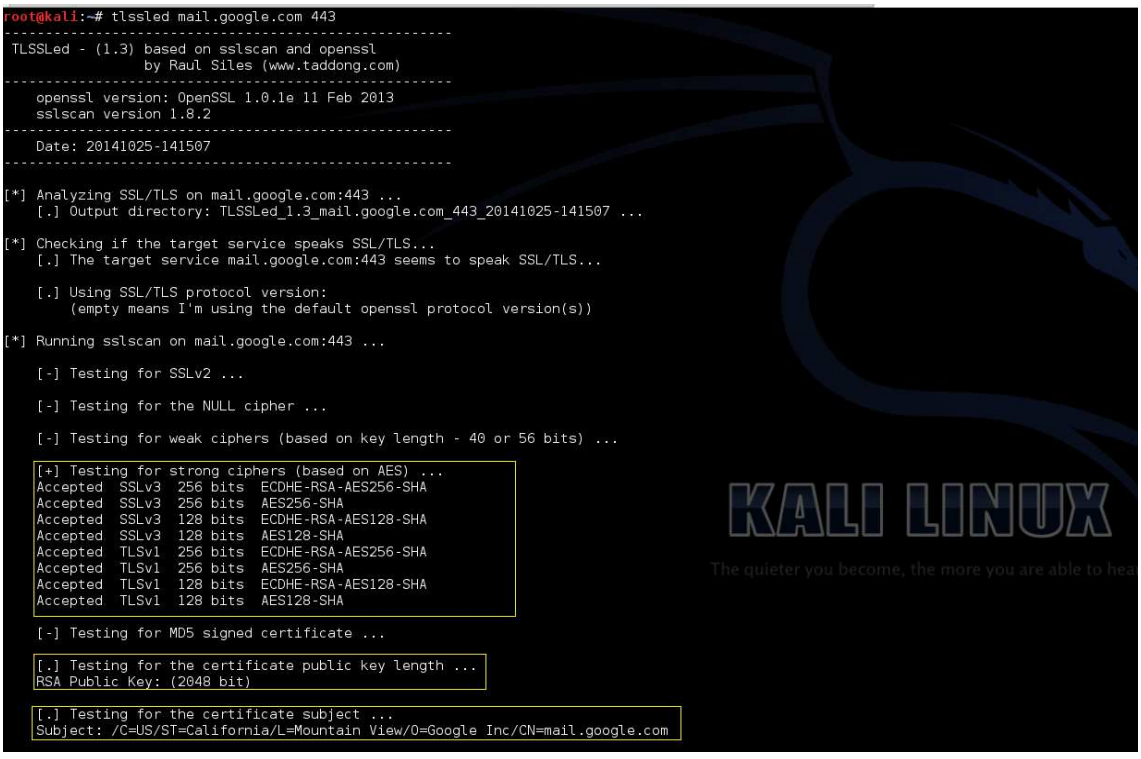


Figura 15: resultado parcial del análisis SSL a un servidor de Google

[24] Podemos ver otro ejemplo seguro con Owasp en la web http://blog.taddong.com/2011/05/tlssled-v10.html

En la imagen, resaltado en amarillo, vemos que el servidor de Google soporta una serie de algoritmos de cifrado fuertes.

5.2.2. SSLDIGGER

Esta herramienta es similar a la anterior, pero para sistemas Windows. También comprueba las versiones instaladas de SSL o la fuerza de los algoritmos de cifrado, aunque es más antigua que TLSSled. Los resultados que obtenemos con ella son los mismos que ya hemos mostrado antes, no aporta nada nuevo, por lo que no los incluiremos en el análisis.

5.3. BALANCEADORES DE CARGA Y WAF

Como ya hemos contado, hay veces en que una aplicación, sea web o no, se encuentra alojada en diversos servidores balanceados. Nuestra tarea aquí será detectar si la aplicación web que estamos auditando está en ésta situación o si, por el contrario, está en un único sistema.

Lo que sabemos hasta ahora de nuestro cliente, es que es una empresa pequeña, y que todas las IPs que hemos recopilado tras aplicar las diferentes herramientas pertenecen al mismo servidor, el mismo que aloja las tres páginas web objetivo. Esto ya nos hace sospechar que en esta ocasión no habrá balanceo de carga. Es más probable que nos encontremos con entornos balanceados cuando auditamos a una empresa de gran tamaño. Sin embargo, el tamaño de la empresa no es un indicador determinante de la existencia o no de un balanceador de carga, por lo tanto, lo recomendable es comprobarlo como un paso más de la fase de mapeo.

Los balanceadores de carga son unos dispositivos que distribuyen el tráfico entre varios servidores web. Estos, en principio, deben ser idénticos ya que, de no serlo, las diferencias existentes podrían indicar la presencia de vulnerabilidades.

La mejor manera de auditar un entorno balanceado es centrar toda la actividad en una sola máquina, no intentar atacar a todas las máquinas que encontremos. Seleccionaríamos uno de los servidores que forman el grupo de balanceo, y dirigiríamos hacia él todos los ataques que realizáramos. Esta forma de actuar se debe a varios motivos:

- **Dejar el menor rastro posible**. Hay ataques que generan ficheros en los sistemas accedidos. Si atacamos a varias máquinas del "cluster", dejaríamos una copia del fichero en cada servidor del grupo. Esto es bastante sucio y no es necesario. Además, aunque como auditores contemos con permiso para hacerlo, debemos intentar actuar como lo haría un atacante malintencionado, y uno con experiencia evitará ir duplicando los rastros que deja en sus víctimas, ya que cuantos menos rastros deje, menos posibilidad habrá de ser detectado.
- **Subir una sola Webshell**. Una "webshell" es un programa web que se utiliza para obtener el control del servidor. En línea con el punto anterior, si dejamos una réplica de un programa en cada servidor, multiplicamos la peligrosidad de que alguien dé con el programa y lo use para ganar el control del sitio. Con una sola réplica es suficiente para trabajar, por lo tanto, no hace falta copiarla en cada máquina.
- **Evitar errores en la detección de vulnerabilidades**. Los balanceadores suelen cambiar el servidor objetivo al que se le envía una petición cada cierto espacio de tiempo. Si no somos cuidadosos a la hora de acceder a un único servidor, puede ocurrir que subamos un fichero a uno de ellos y más tarde pretendamos ejecutarlo en una petición a un servidor distinto, con el resultado de que, obviamente, no funcionará. Si no tenemos la precaución de ceñirnos a una sola máquina, y no nos damos cuenta de cuándo el balanceador nos ha cambiado de objetivo,

entonces podemos cometer el error de pensar que no existe una vulnerabilidad porque la ejecución del fichero no ha funcionado, cuando en realidad, la hemos lanzado a un lugar diferente.

Hay distintos métodos para detectar o identificar balanceadores de carga, por ejemplo: analizar las URLs para ver si son redirigidas a otras distintas; analizar las marcas de tiempo de los servidores para ver si las fechas que devuelven no son coherentes con nuestras pruebas; comparar los valores de última modificación de la página para ver si son distintos; analizar las cookies que establecen los balanceadores para indicar qué servidor se va a usar; ver si hay diferencias en los cifrados SSL soportados.

5.3.1. WAFW00F

La herramienta Wafw00f nos va a servir para averiguar si la web objetivo está en un único servidor, o en un conjunto de ellos. Wafw00f realiza varias peticiones y, utilizando las técnicas anteriormente mencionadas, detecta si hay balanceo de carga, o si hay instalado algún firewall de aplicación, entre otras cosas [29].

Un firewall de aplicación, o WAF (Web Application Firewall) es un tipo de firewall que se encarga de proteger la aplicación web. Este sistema detecta patrones de ataque web y puede parar o modificar estas peticiones. Resulta muy útil detectarlos porque puede interferir en las auditorías que realicemos. Se recomienda que el auditor, al detectarlos, solicite al cliente que los deshabilite durante la auditoría. Esto se hace porque así podremos detectar todas las vulnerabilidades existentes en la aplicación, sin sufrir las interferencias del WAF. Otra posible auditoría es la de verificar las protecciones ofrecidas por el WAF, pero normalmente este tipo de análisis lleva más tiempo que una auditoría de las vulnerabilidades de la aplicación Web.

Aplicamos el siguiente comando al dominio de nuestro cliente:

```
# wafw00f dominio_cliente.com
```

Figura 16: ejemplo de ejecución de Wafw00f

Vemos que la herramienta ha efectuado trece peticiones y no ha detectado ningún WAF, ni tampoco que el servidor esté balanceado, tal como sospechábamos. Ahora ya sabemos que no tendremos que saltarnos ningún WAF.

5.3.2. HALBERD

Similar a la herramienta anterior. Halberd [31] utiliza las técnicas comentadas para descubrir la existencia de un balanceador de carga. Su aplicación sobre el cliente no aporta nada

nuevo al resultado obtenido con "Wafw00f", lógicamente porque no existe balanceador [30]. Pero podemos ver un ejemplo de análisis sobre la web de Google, donde sí existe un balanceador:

```
# halberd www.google.com
```

Figura 17: ejecución de Halberd sobre google.com

5.4 ANÁLISIS DE LA CONFIGURACIÓN DEL SOFTWARE

En este punto ya hemos determinado cuál es el diseño de red de la empresa auditada. El siguiente paso es tratar de entender el sistema que tenemos entre manos a nivel de software. Para ello, vemos cuál es la configuración del sistema operativo, de los servicios de red, y del servidor web. Trataremos de conocer aspectos como qué métodos HTTP soporta el servidor, saber qué características están disponibles, si se soporta PHP o .NET, si el servidor web almacena documentación, o si contiene páginas instaladas por defecto, con scripts o pequeños programas de ejemplos.

5.4.1 NIKTO

La herramienta Nikto[25] es un programa escrito en perl que sirve para buscar este tipo de vulnerabilidades en aplicaciones web. Utiliza una base de datos que contiene los ficheros por defecto de las instalaciones más típicas y los programas vulnerables más conocidos, que suelen estar alojados en los servidores web (como por ejemplo CGIs, ASPs, etc), e intenta acceder a cada uno de ellos, haciendo así una especie de escaneo [32]. Aplicamos la herramienta al servidor del cliente con el siguiente comando:

```
# nikto –host dominio_cliente.com
```

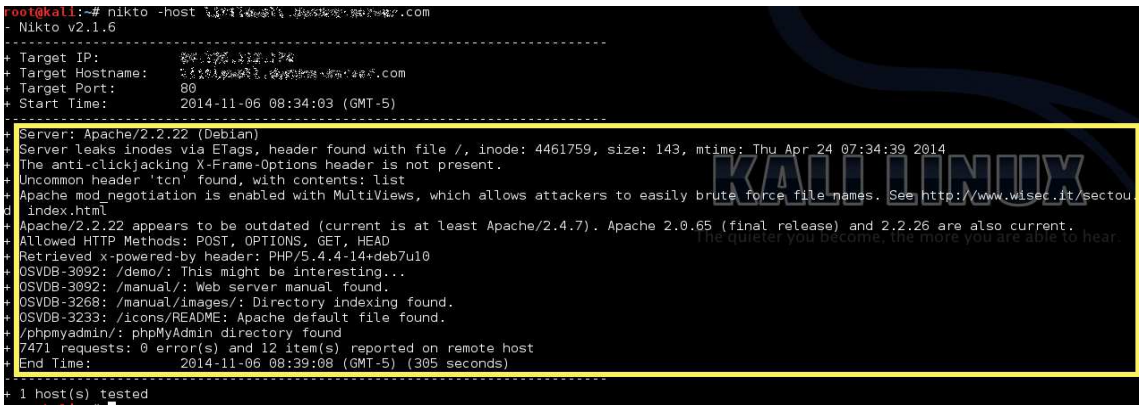


Figura 18: ejecución con Nikto sobre el servidor cliente

[25] Disponible en http://www.cirt.net

Lo que vemos recuadrado en amarillo es toda la información que hemos obtenido de este escaneo, que es bastante. Nos aparecen versiones del software instalado, la lista de métodos HTTP soportados, la presencia de código PHP, algunos directorios y lo que estos contienen, y más cosas. Habrá que volver sobre estos hallazgos más adelante en descubrimiento para cerciorarnos de si hay vulnerabilidades explotables, o se trata de falsas alarmas.

5.5 SPIDERING. NAVEGACIÓN MANUAL.

El último paso de la fase de mapeado es lo que se conoce como Spidering. En inglés, "spider" es "araña", así que la tarea de "spidering" la podríamos traducir como hacer un rastreo por la aplicación, o lo que es lo mismo, una navegación manual en profundidad por toda la web, o aplicación objetivo. La finalidad es conocer a fondo el sitio web e identificar todas las funciones que implemente. Este paso nos llevará el 90% del tiempo de toda la fase de mapeado, porque hay que recorrer y probar toda la funcionalidad de la aplicación, por extensa que sea. Por ejemplo, si el cliente deja subir un archivo, hay que probar a subir archivos; si hay una función administrativa, hay que probar si es accesible sin estar autenticado o si es accesible desde usuarios sin privilegios. Las pruebas completas se realizarán en la siguiente fase de descubrimiento, pero en la fase actual es donde tenemos que encontrarlas, e identificarlas, para volver sobre ello más tarde. El objetivo es señalar posibles funciones críticas o puntos susceptibles de ser vulnerables.

Para entender la diferencia entre esta fase y la siguiente, imaginemos a un ladrón que busca una casa donde colarse a robar. Va paseando por la calle, se fija en los detalles, mira si las ventanas tienen rejas o no, si la puerta es blindada, o si hay sistema de alarma. Una vez detecta una casa fácil con un punto débil, anota la dirección y se va. Esta sería la fase de mapeado. La fase de descubrimiento es cuando el ladrón vuelve por segunda vez con ganzúas o un martillo, preparado ya para tirar la puerta abajo.

5.5.1 ZAP

Para hacer esta navegación a través de la web (en nuestro caso a través de tres webs), vamos a utilizar el proxy ZAP de Owasp[26]. Identificamos objetivos, recursos y parámetros, que luego escanearemos con el escáner y con el fuzzer contenidos en la herramienta.

Para que el proxy ZAP capture todas las peticiones hechas al navegar por la web objetivo, debemos configurar primero el navegador para que coja al ZAP como proxy. En nuestro caso estamos usando el navegador Mozilla Firefox, al que instalamos su plugin Foxyproxy. Lo configuramos para que todo el tráfico web pase a través del ZAP (ver anexo 6). A continuación se muestra la pantalla principal de la herramienta ZAP después de haber navegado un rato por varias webs, entre ellas, el objetivo.

Trabajaremos con varias secciones abiertas. En la sección "Sitios", a la izquierda, se irán listando los distintos servidores que hemos visitado. En la sección derecha podremos ver las peticiones hechas al servidor web, y las correspondientes respuestas de éste, además de los puntos de interrupción en los que podremos capturar parámetros y modificar sus valores antes de enviarlos. Abajo vemos el historial con todas las peticiones realizadas, y también podremos visualizar alertas, escaneos, parámetros, o los procesos de spider y fuzzing [33, 34].

[26] https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project

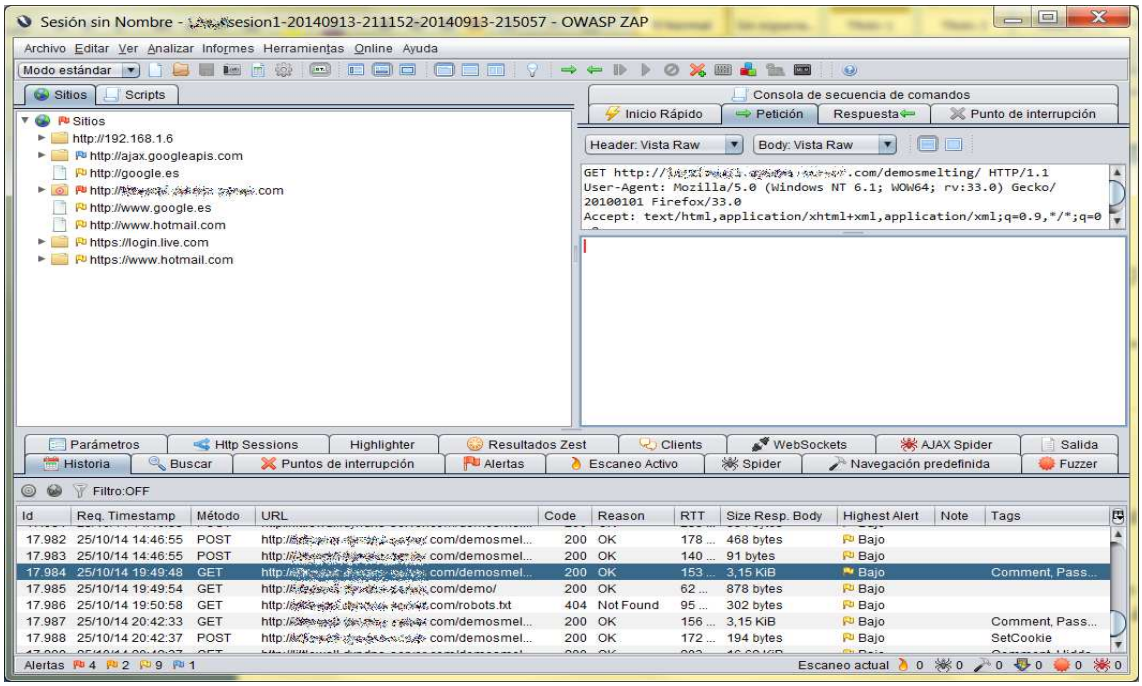


Figura 19: pantalla principal de ZAP en plena recogida de datos

5.6 SPIDERING. ANÁLISIS DE LOS RESULTADOS.

Una vez terminado el recorrido manual de todas las webs, tendremos capturado un árbol con todas las páginas que hemos visitado y empezaremos a analizar los resultados encontrados con la función "Spider" del propio ZAP. El objetivo ahora es buscar ficheros en el servidor web. Esto se hace de forma complementaria al análisis manual, ya que al recorrer la aplicación web a mano podemos dejarnos ficheros sin descubrir. En la pestaña "Spider" de la imagen siguiente se muestran las páginas que está visitando el escáner de la herramienta.

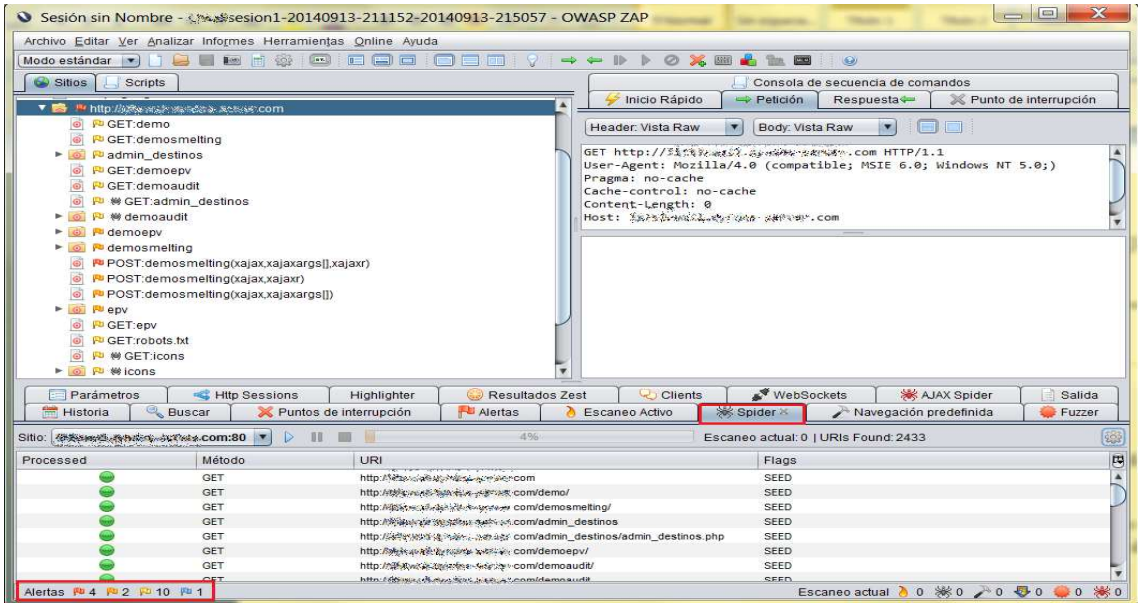


Figura 20: listado de páginas recorridas por el "spider" de ZAP

A partir de este punto entraríamos en la tercera fase de la metodología, la de descubrimiento. En esta fase se volverán a hacer recorridos cada vez que lancemos una herramienta de descubrimiento que no conozca el sitio. Las herramientas actuales no son capaces de leer los recorridos hechos por otras herramientas, es decir, no comparten el conocimiento adquirido. Es habitual, pues, que la tarea de spidering se repita múltiples veces entre esta fase y las siguientes.

CAPÍTULO 6. METODOLOGÍA. FASES DE DESCUBRIMIENTO Y EXPLOTACIÓN

Las dos últimas fases, descubrimiento y explotación, las vamos a explicar conjuntamente en un único capítulo, debido a que están tan interrelacionadas entre ellas que muchas veces es más fácil desarrollarlas en paralelo que secuencialmente por separado. Algunas de las actividades típicas pueden pertenecer tanto a una fase como a la otra. Así que para diferenciarlas, de nuevo entra en acción el criterio del auditor.

Tendremos dos tareas principales a desarrollar: detección y verificación. La detección suele pertenecer a la fase de descubrimiento, y la verificación a la fase de explotación, pero no es estrictamente así en todas las ocasiones. Sin embargo, en este trabajo vamos a seguir este esquema.

Cada una de las dos secciones estará compuesta por los hallazgos concretos que se logren encontrar en esta auditoría. Cada una de las demás auditorías será distinta sobre todo en estas dos fases, ya que los hallazgos serán distintos cada vez, así que las pruebas que habrá que realizar cambiarán. También cambiarán en función de otros factores, como la aplicación a auditar, o las tecnologías que emplee la web. Por ejemplo, hay vulnerabilidades específicas de .NET, de PHP, de Apache, etc.

6.1. DETECCIÓN

En esta sección nos centramos en buscar y detectar vulnerabilidades a partir de los resultados obtenidos durante el mapeado. Se pueden emplear diferentes métodos, y realizar nuevos escaneos con nuevas herramientas que no hayamos usado antes. En nuestro caso, primero veremos cómo se comporta la aplicación ante los errores introducidos a mano adrede, por ejemplo, escribiendo letras o signos en un campo que pide números; después, escanearemos los recursos o métodos que envían parámetros al servidor objetivo; por último, se emplearán ataques de fuzzing sobre los parámetros encontrados en los puntos anteriores.

6.1.1. COMPORTAMIENTO ANTE ERRORES

En esta parte se busca provocar errores en la aplicación para ver cómo se comporta. El tratamiento de los errores es tarea de los programadores de la aplicación, que deben definir qué debe hacer ésta en caso de producirse alguno. Los lenguajes de programación suelen contar con funciones o librerías específicas para realizar un manejo de errores, o excepciones, adecuado. Por ejemplo, Java o C++ usan bloques "try – catch". Así, el programador puede controlar la respuesta del sistema ante errores, o bien retornando un mensaje de error, o devolviendo un valor por defecto, o ejecutando un fragmento de código alternativo al del funcionamiento esperado. Si no se define la respuesta ante errores, el programa podría mostrar por el navegador lo que ha sucedido en el servidor, lo cual no interesa.

Según los recursos que encontremos en las webs, las pruebas que haremos variarán. En este caso, no encontramos recursos explotables en la segunda web (escuelatest.com), ni en la tercera (auditandowebs.com), pero sí en la primera (fusionmetales.com). Viajamos a mano hasta una URL con campos para hacer pedidos nuevos. Uno de ellos nos pide introducir el peso del metal que queremos pedir. En su lugar, vamos a introducir una cadena de cuatro letras "aaaa":

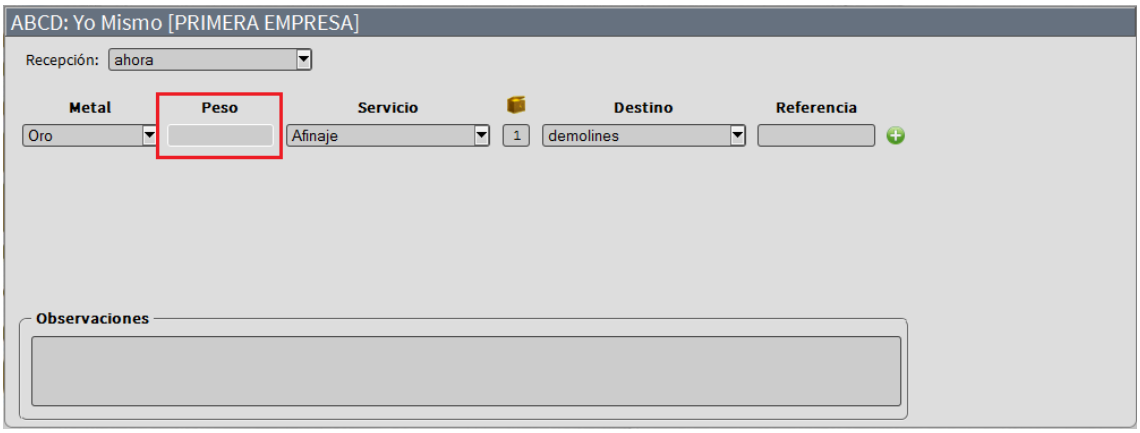


Figura 21: tabla para realizar pedidos de metales con el campo a analizar

Como la web no permite insertar más que números, capturamos la petición con el proxy ZAP, ponemos un punto de interrupción para todas las peticiones, y cambiamos a mano el valor del parámetro "peso" en el payload, por "aaaa": %3Epeso%3C%2Fk%3E%3Cv%3Eaaaa%3

Al enviar el pedido, la web no avisa del error, sino que recoge la cadena de aes. La aplicación no prevé la posibilidad de que un usuario sepa introducir caracteres no permitidos, por lo que no da una respuesta controlada por el desarrollador. Además, envía información sensible que puede ser capturada. En la imagen siguiente, vemos la respuesta generada, donde dos "warning" nos indican una serie de datos: nombres de funciones de la base de datos (`pg_query`), nombres de tablas existentes (`encargos_procesado`), valores de otros parámetros (`10, 1, 7, aaaa`…), e incluso rutas de directorios del servidor, que no deberíamos conocer:

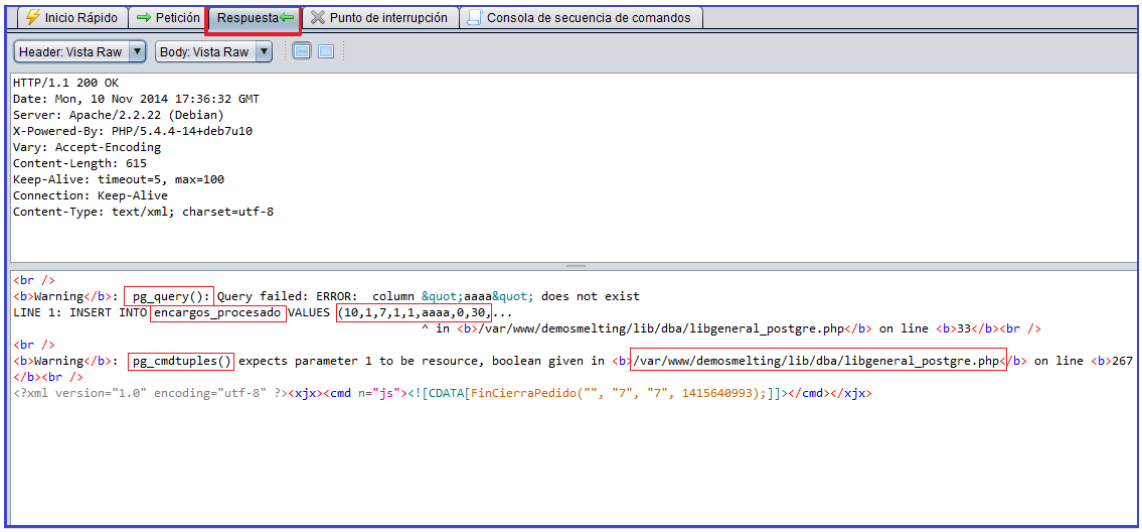


Figura 22: captura de la respuesta ante el error

En la misma web hay otros formularios vulnerables. Por ejemplo, nos damos de alta como cliente ficticio en el apartado de pre-clientes; tras validar el registro, en una pestaña de datos, encontramos un campo llamado "tarifas cliente". Al clicar sobre él, se abre una ventana donde se listan los precios de los servicios que se pueden solicitar. En principio, ya es un error que la aplicación nos deje cambiar los precios sin haber entrado como administradores:

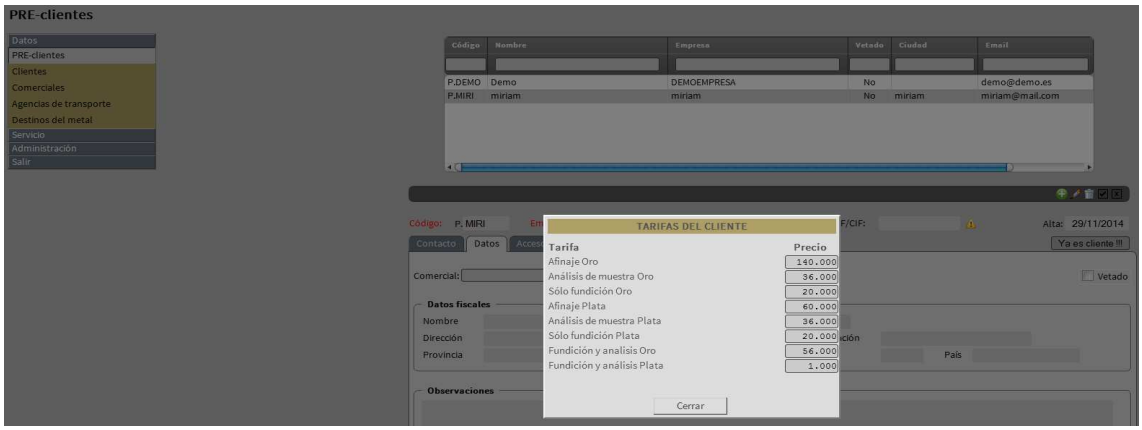


Figura 23: ventana de tarifas emergente tras darnos de alta como nuevo cliente

En la imagen vemos cómo se ha podido cambiar a mano el precio del último servicio: "fundición y análisis plata" tenía un coste igual al servicio anterior con oro, o sea, de 56.000 euros, pero lo hemos cambiado por sólo 1.000. El cliente no debería poder cambiar sus propias tarifas a menos que tenga cuenta con permisos de administrador. Este es un fallo menor, pero que nos da pie a detectar vulnerabilidades más peligrosas. Tal como hicimos antes, capturamos la petición en el ZAP, e introducimos una letra "a" en el precio antes de enviarlo al servidor. En lugar de "1.000", enviamos "a.000":

```
xajax=GuardaTarifasCliente&xajaxargs[]=%3Eprecio%3C%2Fk%3E%3Cv%3Ea.000%3C
```

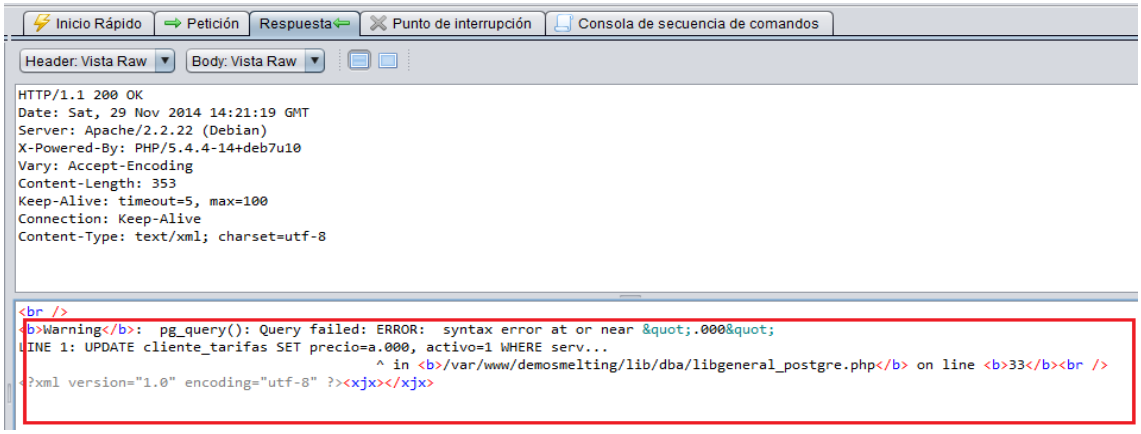


Figura 24: captura de la respuesta ante el envío del precio erróneo

Como vemos encuadrado en rojo, la letra ha generado un error de sintaxis que no se ha tratado, por lo que podemos ver nombres de funciones, tablas de la base de datos, y rutas del servidor.

En conclusión, hemos detectado que la captura y tratamiento de errores no es correcta, da lugar a una vulnerabilidad que hemos explotado a mano, con ayuda del proxy ZAP, al enviar datos inválidos en campos de formularios. Como dijimos al principio, descubrimiento y explotación han ido en paralelo. En el informe final, este fallo deberá figurar como vulnerabilidad de riesgo medio.

6.1.2. ESCANEO DE LAS VULNERABILIDADES

En este punto hemos de lanzar diversos escaneos de ataque contra todos los recursos con parámetros, excepto contra las funcionalidades peligrosas que podrían saturar el servidor, por ejemplo, una funcionalidad que sirva para enviar correos, podría inundar la cuenta de correo de destino con envíos masivos. En estos casos delicados es mejor hacer las pruebas a mano y evitar el escaneo automático.

Una vez lanzados los escaneos de ataque contra las tres webs que estamos auditando, repasaremos los resultados de éstos almacenados en el historial de ZAP, que puede verse en la sección transversal de abajo, en la primera pestaña:

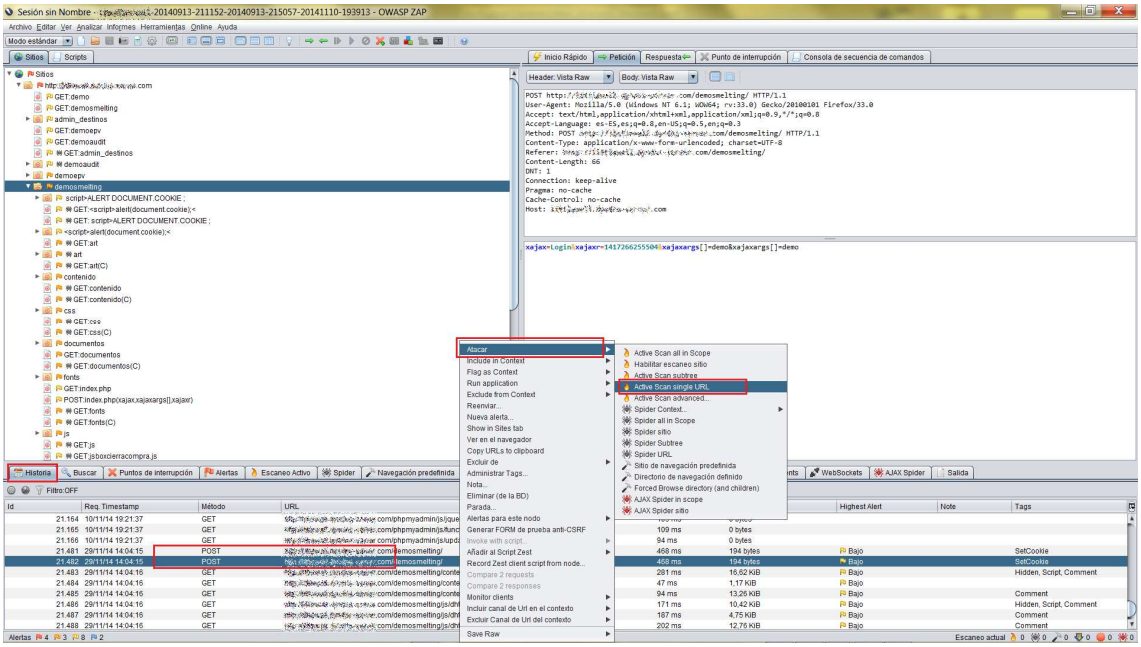


Figura 25: repaso del historial en busca de métodos críticos

Sabemos que el uso público de algunos métodos HTTP, tales como PUT o DELETE, puede ser peligroso si el servidor web no está bien configurado[35]. Sabemos también, por el resultado que nos dio la herramienta "Nikto" en la sección anterior[27], que el servidor que estamos auditando permite los métodos siguientes: POST, OPTIONS, GET, HEAD. Así que, miramos en el historial de ZAP qué métodos nos aparecen, y encontramos sólo dos: GET y POST. De ambos, seleccionamos las URLs que tengan parámetros en sus peticiones. Para lanzar los ataques, tal como se ve en la figura 25, nos colocamos sobre las URLs de interés, y con clic derecho del ratón sobre una de ellas, lanzamos ataques de diferentes tipos. En la imagen, recuadrado en rojo vemos el ataque seleccionado: "active scan single URL".

Si seleccionamos la pestaña "Alertas", vemos que nos va listando lo que detecten los escaneos, ordenado por nivel de riesgo. Al lado de cada alerta vemos banderines de colores. Si es rojo, significa que el riesgo de ese hallazgo es alto. Si el banderín es naranja, el riesgo es de grado medio, y si es amarillo, bajo. Los banderines azules son informativos. En la siguiente

[27] Ver figura 18, sección 5.4.1.

imagen podemos ver un total de ocho alertas rojas de alta prioridad, cuatro de riesgo medio, nueve de riesgo bajo, y cuatro alertas informativas:

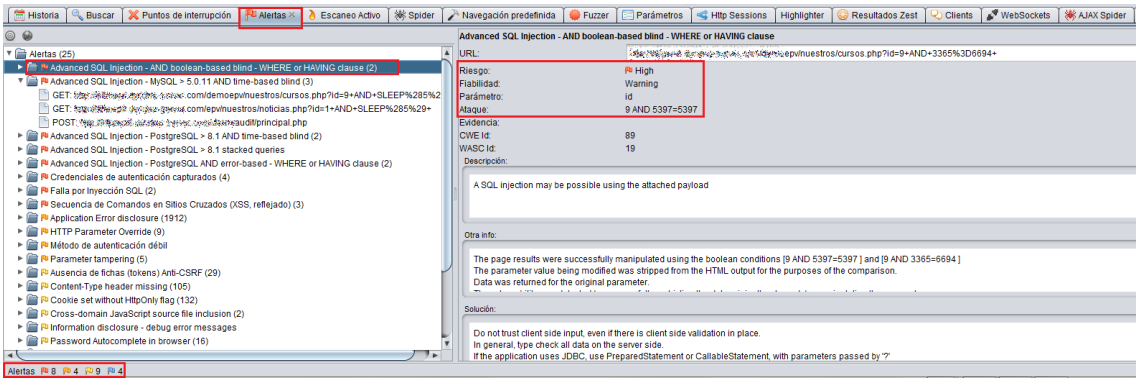


Figura 26: pestaña de alertas encontradas tras los ataques

El riesgo va asociado a la prioridad, así que los análisis deben comenzar por el principio de la lista: las alertas rojas. Con cada alerta deberemos verificar, en el siguiente apartado, si realmente es explotable o no lo es. Pueden darse falsos positivos con los escaneos automáticos, por eso, el auditor debe asegurarse a mano de que cada hallazgo es realmente una vulnerabilidad, y no darlo por hecho.

Si seleccionamos la primera alerta con el ratón, a la parte derecha aparece información asociada a ese hallazgo: la URL, el riesgo, el ataque lanzado, una descripción, etc. Aunque algunas alertas se llamen igual o sean del mismo tipo (como las primeras cinco "SQL injection"), se refieren a vulnerabilidades distintas, encontradas en páginas distintas o en escaneos distintos. Por eso, no debemos agrupar por nombre, sino considerar cada alerta por separado. Al final de cada una, vemos un número entre paréntesis. Esto indica en cuántos lugares se ha detectado la alerta. Si desplegamos una alerta, por ejemplo la segunda que tiene un 3 entre paréntesis, vemos que este aviso se ha dado en dos peticiones GET y una POST, tres en total. En este caso, sólo hemos hecho la tarea de descubrimiento. En el punto de verificación se hará la parte de explotación de cada uno de los veinticinco hallazgos.

6.1.3. FUZZING DE PARÁMETROS

De forma adicional al escaneo de vulnerabilidades de ZAP se utiliza "fuzzing" de parámetros. Este tipo de pruebas consiste en enviar, en los parámetros de la aplicación, múltiples valores con la intención de provocar errores en la misma. Se pueden encontrar vulnerabilidades que no se encuentran con el escaneo sencillo, ya sean de inyección SQL, inyección de comandos, XSS, errores no capturados o denegación de servicio.

Para la realización de estas pruebas se utilizan varios diccionarios. ZAP lleva integrada la función de "fuzzing", así que lo usaremos de nuevo. Ésta función incluye dos colecciones de diccionarios (fuzzdb[28] y jbrofuzz[29]) que contienen listas de usuarios y contraseñas, patrones conocidos de ataque agrupados por familias, e incluso, rutas por defecto de distintos productos conocidos. En la imagen vemos dos ejemplos de "fuzz" aplicados sobre el mismo parámetro:

[28] https://code.google.com/p/fuzzdb/
[29] https://www.owasp.org/index.php/JBroFuzz

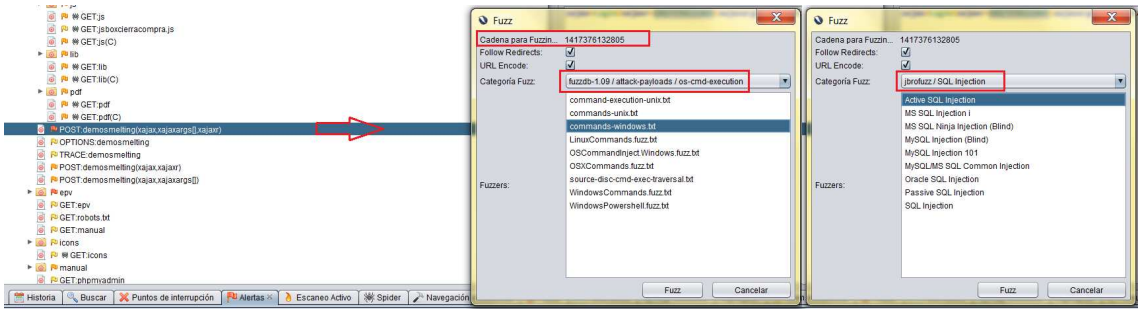


Figura 27: selección de una cadena para fuzzear con los dos diccionarios nombrados

Como decíamos, buscamos una URL que contenga algún parámetro dinámico que se vaya a enviar a la aplicación web, y por tanto al servidor que la aloja. Seleccionamos con el ratón su contenido (el valor que vamos a "fuzzear"), y con clic derecho del ratón, abrimos el menú que se ve en la imagen siguiente, y elegimos la opción "fuzz". Esto abre otra ventana donde deberemos elegir el/los diccionarios que se quiere emplear. Los resultados de este paso se explotarán en el siguiente punto:

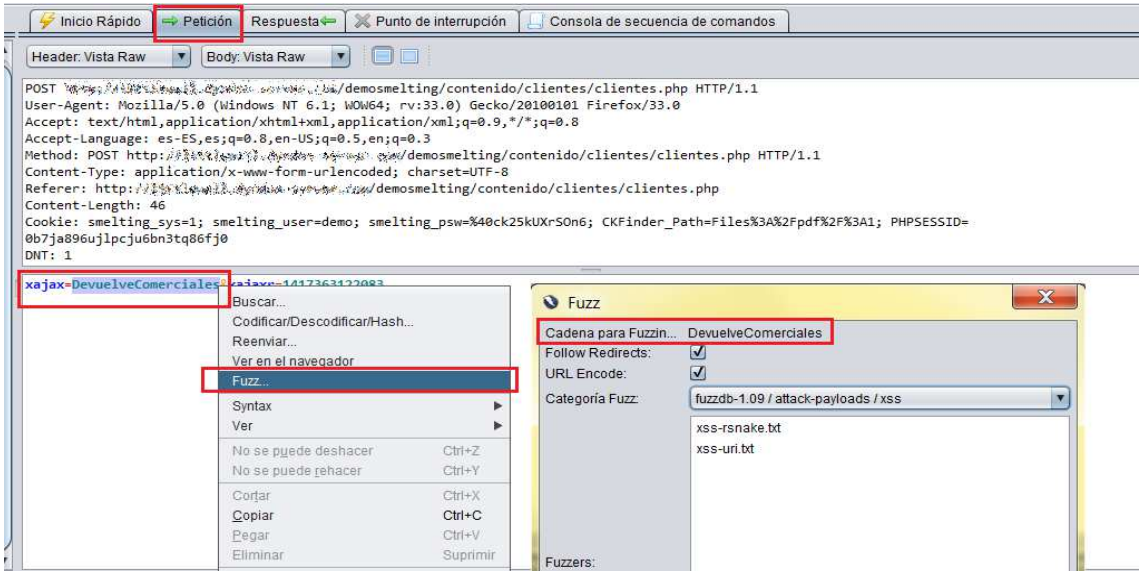


Figura 28: selección del valor de un parámetro a fuzzear

6.1.4. DIVULGACIÓN DE INFORMACIÓN

Durante la fase de mapeado se detectaron secciones en la web que permitían al usuario descargarse facturas en formato .pdf. Con nuestro usuario registrado hemos ejecutado esta funcionalidad, y hemos descubierto que la creación del documento no se hace correctamente. En él, aparecen datos privados que los clientes no deberían poder ver, por ejemplo, rutas absolutas de ficheros dentro del servidor web. Este fallo proporciona información al atacante, que, sin duda alguna, también se creará un usuario cliente y, por supuesto, probará a conciencia cualquier funcionalidad que le permita crear y descargar documentos. Añadiremos este hallazgo al informe como vulnerabilidad de riesgo bajo.

En la siguiente imagen podemos ver la parte del documento donde se divulgan datos por error, algunas rutas absolutas que hemos emborronado por confidencialidad:

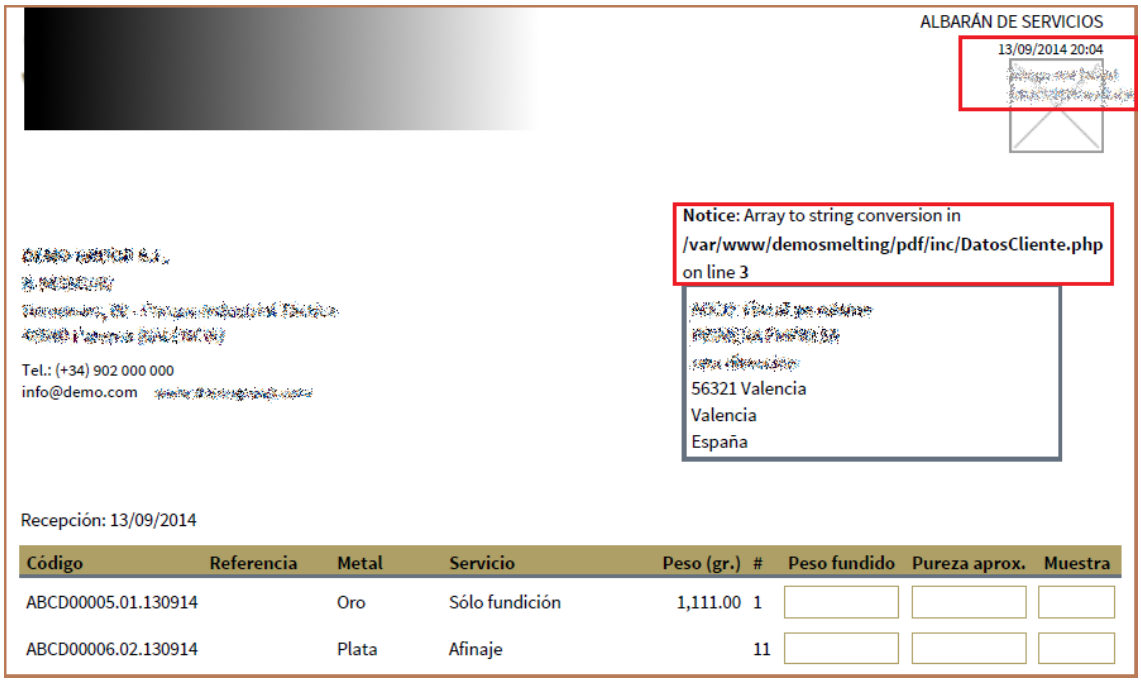


Figura 29: cabecera del documento pdf descargado del servidor

6.2. VERIFICACIÓN MANUAL

Una vez realizados todos los descubrimientos posibles, comenzamos con la verificación manual de cada hallazgo sospechoso que aún no haya sido confirmado. Además de las alertas encontradas por ZAP, hay otras encontradas durante el recorrido manual por la web, por ejemplo, en la fase de mapeado se detectó una funcionalidad de subida de ficheros que suele entrañar riesgos de seguridad. Vamos a revisarlas una a una.

6.2.1. SUBIDA DE FICHEROS

En la fase de mapeado detectamos alguna funcionalidad de subida de ficheros. Al clicar en "DNI" o cualquier otro campo de esta sección, se abre una ventana que permite subir un fichero a la web. Debería permitir sólo formatos de texto (.txt, .doc, .pdf, etc), o formatos de imagen (.jpg, .png, .bmp, etc), pero ningún ejecutable ni otros formatos desconocidos.

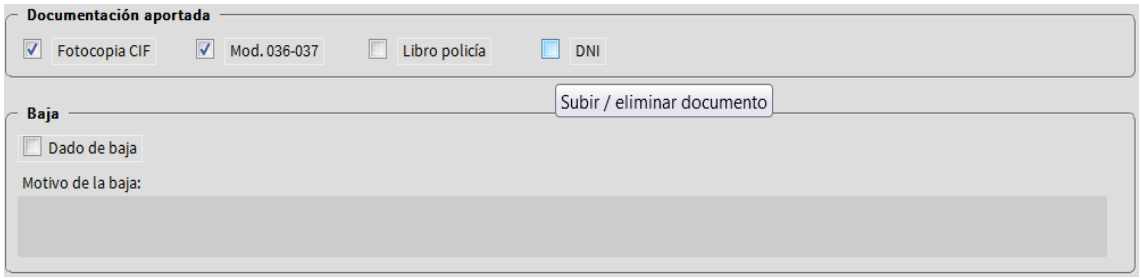


Figura 30: sección de la web que podría ser vulnerable

Ahora vamos a probar si estos campos son vulnerables. Utilizamos los ficheros de prueba de antivirus Eicar [36], que son inocuos, pero que deberían ser detectados como virus por

cualquier antivirus[30]. Con esta prueba buscamos verificar si la aplicación web hace las comprobaciones sobre los ficheros subidos por los usuarios para ver si son maliciosos o no.

Efectivamente, conseguimos subir el fichero "eicar.com" como si fuera el DNI y la web lo almacena. En la imagen vemos el directorio donde hemos subido el fichero binario:

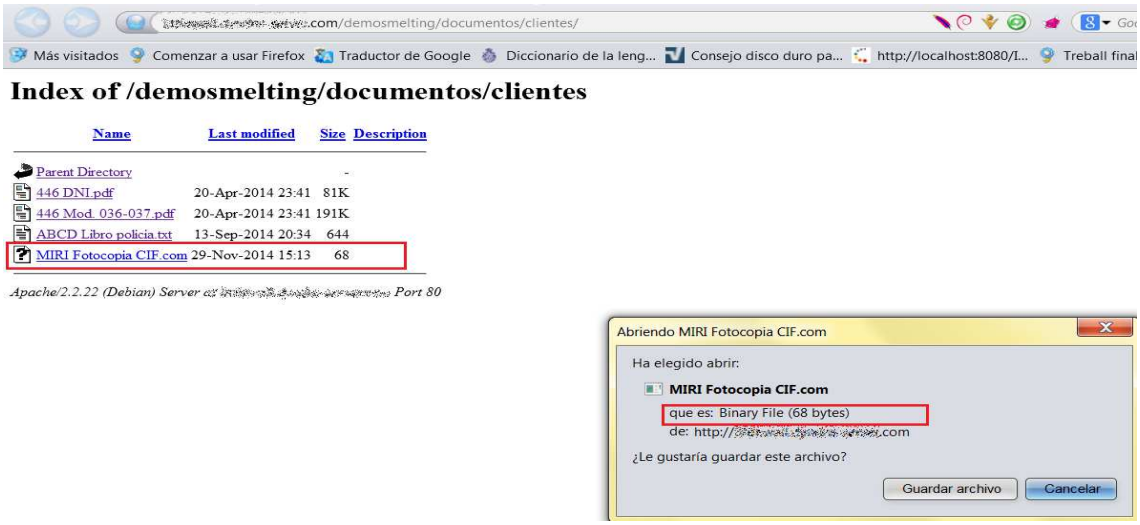


Figura 31: descarga de eicar previamente subido al directorio /documentos/clientes

Hemos explotado con éxito otro fallo que deberá documentarse en el informe como riesgo alto. A raíz de este fallo, vamos a comprobar si a parte de los ficheros "Eicar" también podemos subir una webshell en PHP, con el objetivo de controlar el servidor desde ella.

Figura 32: parent directory actualizado tras subir tres webshells al servidor

Recuadrado en rojo vemos varias webshells que hemos logrado subir. Comprobamos cuál funciona, clicando sobre el link. Una de ellas (MIRI-shell.php) nos permite el acceso al servidor Linux, con permisos de usuario `[www-data]` (lo vemos recuadrado en la franja azul).

Este usuario es el que usualmente se emplea para ejecutar el servidor web Apache:

[30] http://www.eicar.org/85-0-Download.html

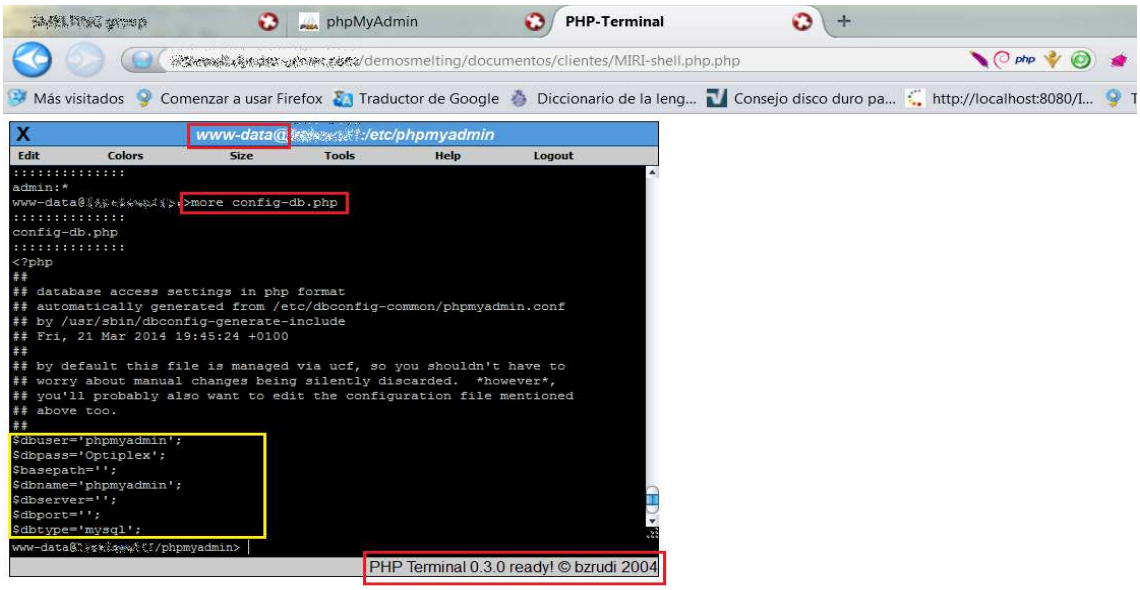


Figura 33: terminal de linux abierto por web donde podemos ejecutar comandos

En la imagen vemos la versión del terminal, y el comando que hemos ejecutado para ver el contenido del fichero "config-db.php": `> more config-db.php`. El resultado es la impresión del fichero por pantalla. En amarillo señalamos toda la información sensible que se ha averiguado. Podemos seguir navegando y editando ficheros delicados para conseguir más información.

Respecto a la segunda web (escuelatest.com), también encontramos una sección que permite subir ficheros, así que procedemos a comprobar si el fallo es exclusivo de la primera web o si también afecta a la segunda. Para ello, cargamos la ruta en el navegador, y en la función de "añadir" ficheros, probamos a subir, por ejemplo: "eicar_com.txt". Tal como se ve en la imagen, la prueba resulta exitosa. Por tanto, esta web es también vulnerable.

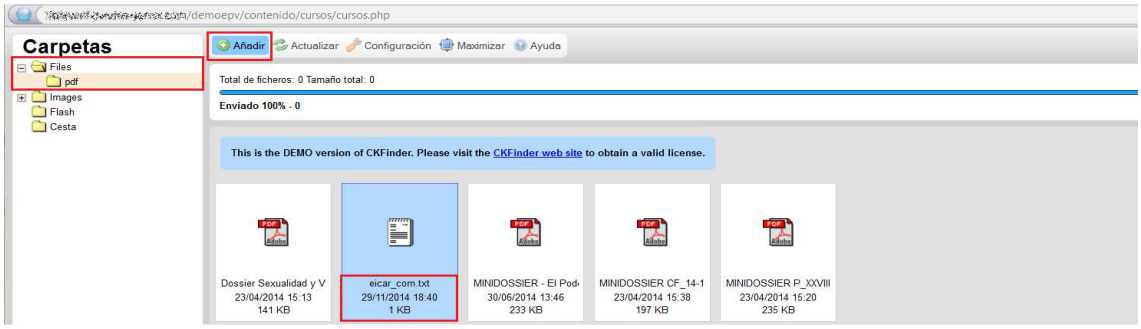


Figura 34: fichero "eicar" añadido a la carpeta de cursos

6.2.2. INYECCIÓN SQL

Entre las alertas listadas por ZAP hay por lo menos cinco inyecciones SQL que habremos de verificar. Todas ellas son alertas de riesgo elevado [8, 38, 39].

6.2.2.1. CASO 1: INYECCIÓN SQL EN PÁGINA DE LOGIN

En la aplicación web "fusiónmetales.com" tenemos que identificarnos como usuario para entrar. Como ZAP indica que el login puede ser vulnerable, vamos a verificarlo.

Figura 35: página de acceso para usuarios por verificar

Lo explotaremos de dos formas: primero, accediendo sin credenciales, saltándonos así la condición de usuario registrado legítimo, y segundo, extrayendo información con SQLmap.

1. **Bypass del login de la aplicación web**.

Se llama "bypass" a la acción de entrar en la aplicación sin disponer de credenciales. Esto ocurre cuando la aplicación web envía los parámetros de usuario y contraseña, a través del formulario de "login", sin emplear ningún tipo de filtrado sobre los mismos. Todo formulario, al recibir los parámetros, los concatena en una cadena que envía a la base de datos SQL como consulta.

Por ejemplo, una consulta típica podría ser:

```
SELECT * FROM users WHERE usuario='$usu' and password='$pass'
```

Si en lugar de insertar en la consulta los parámetros esperados, insertamos órdenes de código que la aplicación interpreta, y además ejecuta, entonces estamos ante un fallo. Las pruebas de SQL estándar consisten en decir a la aplicación que se ejecute siempre que una condición sea verdad. Por ejemplo, insertamos en el campo "usuario" la siguiente orden: **' OR 1=1 --**.

El resultado de la consulta enviada a la base de datos sería:

```
SELECT * FROM users WHERE usuario='' OR 1=1-- and password=''
```

Esta consulta se cumple siempre que haya un usuario válido, o bien, que uno sea igual a uno. Como uno siempre es igual a uno, da igual que el usuario no sea válido; la sentencia siempre será verdad. Como además se interpreta que lo que va detrás de dos guiones es un comentario, la consulta consigue que se ignore la condición "and password". En conclusión, hemos forzado un resultado verdadero y, por lo tanto, podremos entrar en la aplicación sin conocer ni el usuario ni la contraseña, a menos que la aplicación no sea vulnerable a inyección. Procedemos a comprobarlo clicando "acceder":

Figura 36: orden SQL insertada en el campo vulnerable

Efectivamente hemos ganado acceso, deberemos reportarlo en el informe como riesgo alto:

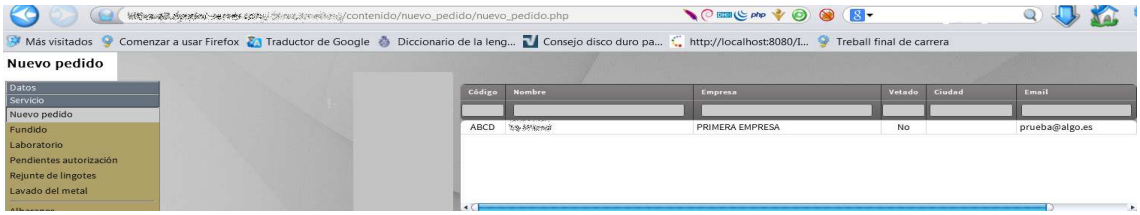


Figura 37: pantalla nuevo pedido a la que hemos podido acceder por medio de la inyección

2. **Extracción de información con SQLMAP**.

Tras confirmar la vulnerabilidad anterior, vamos a explotarla tal como haría un hacker malicioso, extrayendo la máxima información posible de la base de datos mediante la herramienta

SQLMAP [37]. Esta herramienta, de código abierto, escrita en Python, es una de las más conocidas para realizar pruebas de penetración. La ventaja que aporta es que automatiza el proceso de detección y explotación de todo tipo de fallos de inyección SQL. Se encarga de realizar peticiones de diferentes tipos (GET, POST, etc) a los parámetros de las URLs vulnerables que se le indique. La herramienta está disponible en la página web oficial[31]. Para usarla, debemos arrancar la máquina virtual Kali Linux, donde la tenemos instalada. En un terminal, ejecutamos un comando "sqlmap" indicando las opciones que nos interesen, por ejemplo, la URL objetivo completa con [–u], el parámetro vulnerable con [–p], la cadena de datos que enviamos por POST con [--data], e incluso le podemos pedir que nos enumere las tablas de la base de datos y todas sus columnas con [--table] o [--columns]. Por ejemplo:

```
# sqlmap –u página_vulnerable.php --data="cadena_login_pass" -p "parámetro"
```

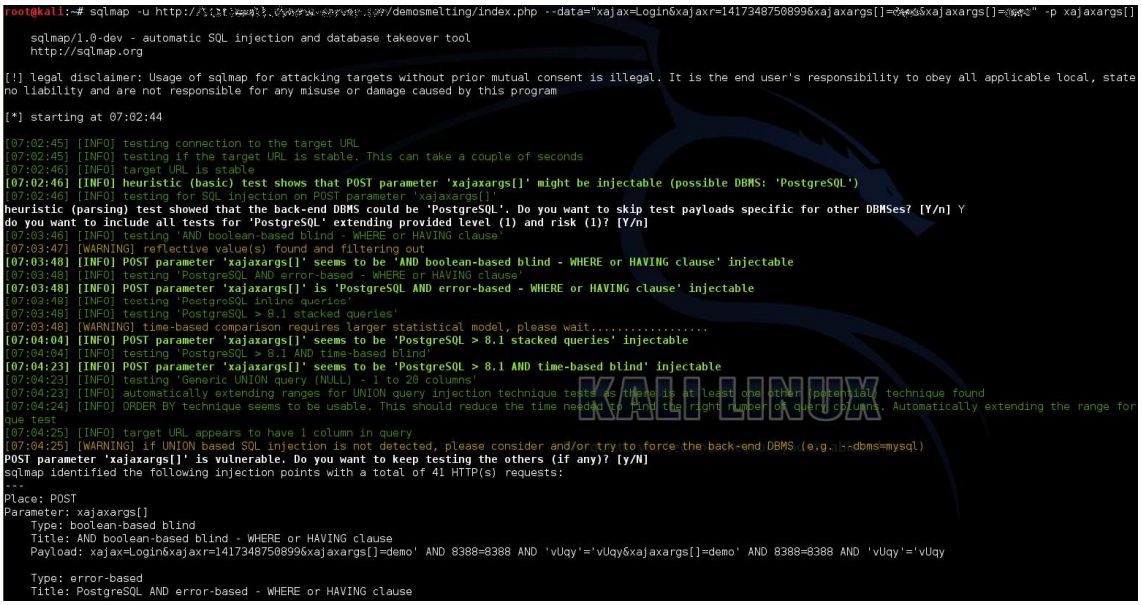


Figura 38: ejecución de la herramienta sqlmap

Si la aplicación es vulnerable, sqlmap extraerá gran cantidad de información de las tablas, como los usuarios registrados, el nombre de las variables, el banner de la base de datos, o incluso se puede hacer un volcado completo de la base de datos. En las siguientes imágenes vemos un ejemplo de lo que hemos conseguido extraer:

Figura 39: muestra de dos BDs encontradas

[31] http://sqlmap.org/

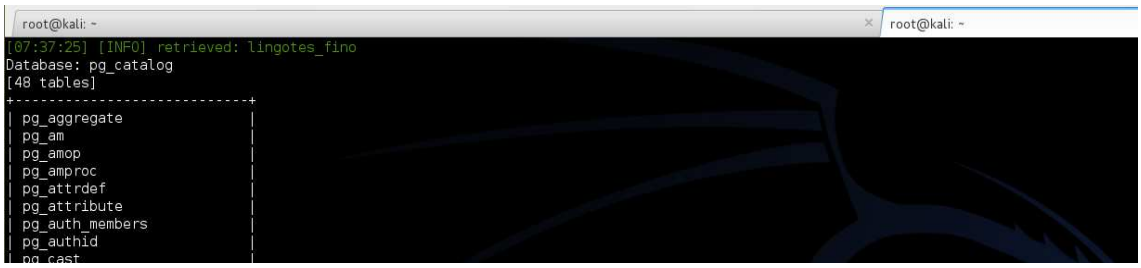


Figura 40: muestra de parte de la tabla encontradas de la BD pg_catalog

6.2.2.2: CASO 2: INYECCIÓN SQL CIEGA

Éste es otro tipo de inyección, encontrada en la segunda web (escuelatest.com). Se llama ciega porque no se sabe el resultado de la operación, al no ser éste visible, haya un error en la base de datos o no. La página web no devuelve errores de ejecución de la base de datos. Así que no podemos verificarlo como antes, ya que aunque introduzcamos una cadena y aunque el servidor la acepte, no podremos ver el resultado de la misma. Sin embargo, sí detectamos un cambio en la página mostrada al cambiar de una cadena verdadera a una falsa. ZAP nos indica la URL vulnerable, acabada en [`cursos.php?id=9`], así que la cargamos en el navegador y añadimos al final un 1=1 codificado en codificación URL. Nos queda semejante a:

`http://dominio_cliente.com/ejemplo/nuestros/cursos.php?id=9+AND+1%3D1+`

La página se carga con normalidad, no advertimos nada. Ahora repetimos lo mismo pero añadiendo un 1=2 para que el resultado sea falso. Lo que nos aparece es la página pero esta vez mostrando un error, señal de que ha ejecutado el código inyectado en la base de datos, por lo tanto es vulnerable, y su riesgo asociado es alto:

`http://dominio_cliente.com/ejemplo/nuestros/cursos.php?id=9+AND+1%3D2+`

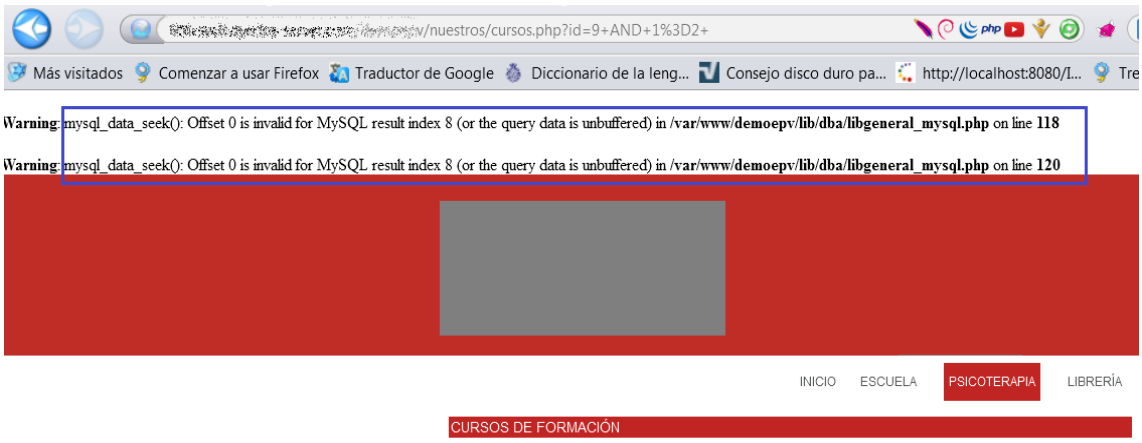


Figura 41: resultado de insertar un código SQL falso en la segunda web

Podemos volver a usar SQLmap para intentar sacar las tablas de la base de datos de la segunda página web. Como es lógico, la prueba ha resultado exitosa, tal como vemos en la siguiente imagen, donde se muestran dos de las tablas de la segunda base de datos: noticias y cursos. Además vemos el número de columnas, el nombre de las variables y el tipo.

Figura 42: algunas tablas de la segunda página web

6.2.3. CROSS-SITE SCRIPTING (XSS)

El escáner de ZAP detectó en la tercera web (auditandowebs.com), y en la primera, tres vulnerabilidades XSS de tipo reflejado; esto es que la web las devuelve al navegador pero no las almacena. Ahora procedemos a verificarlas. Para realizar la verificación, enviamos a mano ataques de scripting a los recursos detectados como vulnerables y observamos el resultado. En caso de ser vulnerables, veremos la ejecución del script por medio de una ventana emergente que se abrirá en el navegador con algún tipo de mensaje [40].

Por ejemplo, dos de los scripts más usados para hacer estas comprobaciones son:

- `<script>alert('XSS')</script>`

  Este script abrirá una ventana con el texto "XSS".

- `<script>alert(document.cookie)</script>`

  Este script abrirá una ventana con el valor de las cookies vigentes en el navegador.

En el manual de pruebas de OWASP se pueden encontrar listas exhaustivas de este tipo de ataques y también están recogidos en la web de "Owasp Cheat Sheets", en el apartado de XSS[32]. En nuestro caso, vamos a usar un ataque sencillo para las pruebas. Abrimos en el navegador las tres URLs que el escáner ha detectado como vulnerables. La primera es la función de "login" de la primera web (fusionmetales.com) que veíamos en la figura 36. En la alerta de ZAP se muestra en el segundo lugar como método POST (en la imagen, con fondo azul):

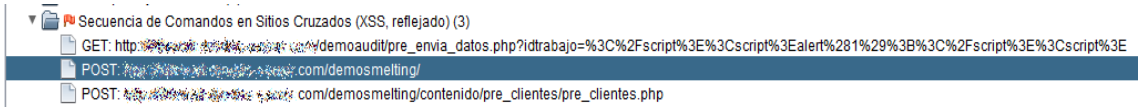


Figura 43: alerta XSS de ZAP desplegada para ver el contenido

Vemos que en este caso, hay ruta pero no hay parámetro. Así que nos logamos. Usamos ZAP para capturar los parámetros del login, poniendo un punto de interrupción antes del envío, y cambiamos el valor del parámetro de la petición (`xajax`), por el ataque:

```
xajax=</cmd><script>alert(1);</script><cmd>
```

[32] https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet

Si el ataque funciona, deberemos ver una ventana con el número "1" impreso. La imagen siguiente muestra el resultado:

Figura 44: ventana emergente que muestra el ataque XSS fallido

El ataque no ha tenido éxito, ya que en lugar del uno, vemos toda la cadena de script. El motivo lo vemos en la respuesta del ZAP: el script se ha enviado como parte de un comentario, por lo que no se ejecuta. Al no ejecutarse se ha evitado el peligro. Aún así, el hallazgo y la prueba deben documentarse en el informe de resultados, indicando que no ha sido exitoso.

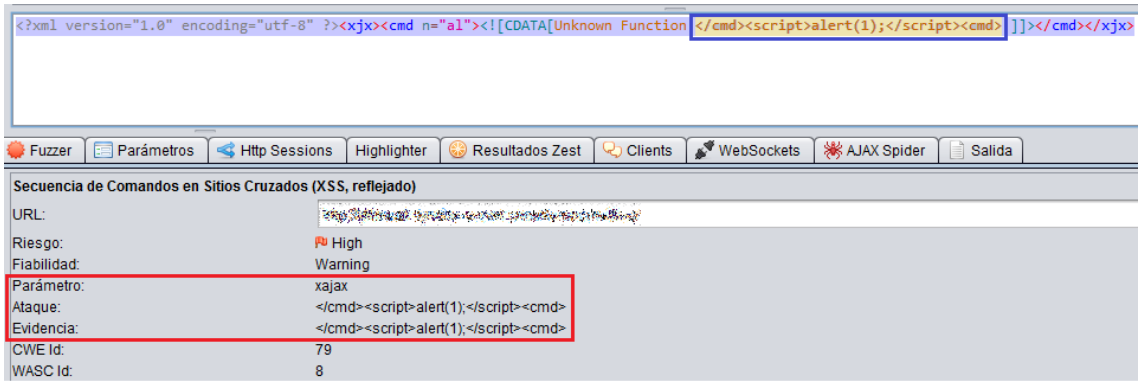


Figura 45: script de ataque como comentario en el código xml

La segunda URL que detecta ZAP es igual a esta: no vulnerable porque el script de ataque se envía como comentario, no como valor de parámetro. Pasamos a analizar el tercer caso. Cargamos la tercera web (indicada en la figura 43 precedida del método GET). El parámetro a atacar es "idtrabajo". Lo igualamos al código que creará la ventana de alerta:

```
idtrabajo=</script><script>alert(document.cookie);</script><script>
```

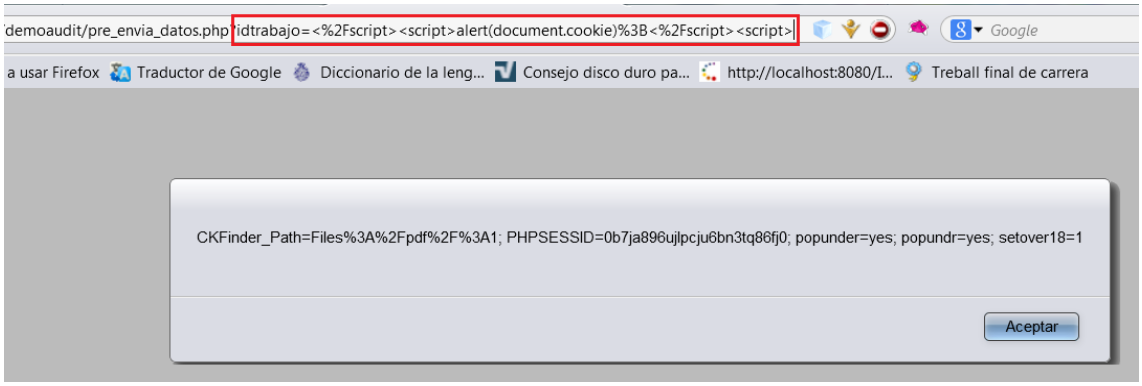


Figura 46: ataque XSS realizado con éxito al parámetro "idtrabajo"

En la imagen vemos cómo el ataque ha tenido éxito, cómo se pudieron obtener las cookies del navegador, y por tanto, la tercera web es vulnerable a XSS [41, 42, 43].

Lo reportaremos en el informe final como vulnerabilidad con riesgo de nivel alto.

6.2.4 LISTADO DE DIRECTORIOS

Durante la fase de mapeado observamos a través de la herramienta ZAP que la aplicación subía los ficheros de clientes a un directorio concreto que no aparece al recorrer la web a mano. Vamos a descubrir si es posible acceder a esa ruta, u otras, sin seguir el flujo natural de navegación, lo cual supondría una vulnerabilidad. La prueba es fácil, y sirve tanto para hacer el descubrimiento como para la explotación [44, 45].

En la imagen siguiente, vemos la petición de ZAP donde hemos dado con la ruta sospechosa (recuadrada en rojo). Con fondo violeta destacamos la ruta visitada con el navegador, que nos permite subir ficheros:

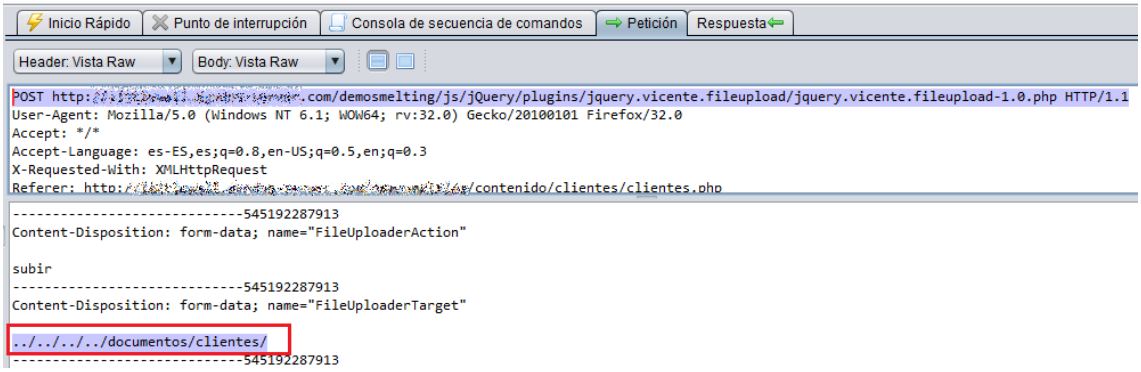


Figura 47: posibles directorios ocultos descubiertos

En el navegador, retrocedemos cuatro directorios de la ruta cargada, y después añadimos los dos nuevos: [`documentos/clientes/`]. Obtenemos el siguiente resultado:

Figura 48: acceso indebido a un listado de directorios privado

No sólo podemos acceder al listado de directorios sino que además, clicando en cualquier fichero, podemos abrirlo y descargarlo. La herramienta Nikto (figura 18) también nos sugirió algunos directorios que podían ser accesibles. Ahora es el momento de probarlos. En la barra de navegación vamos introduciendo a mano todas las rutas, y así conseguimos acceder a:

- http://fusionmetales.com/demosmelting/documentos/clientes/
- http://fusionmetales.com/manual/
- http://fusionmetales.com/manual/images/
- http://fusionmetales.com/icons/README/

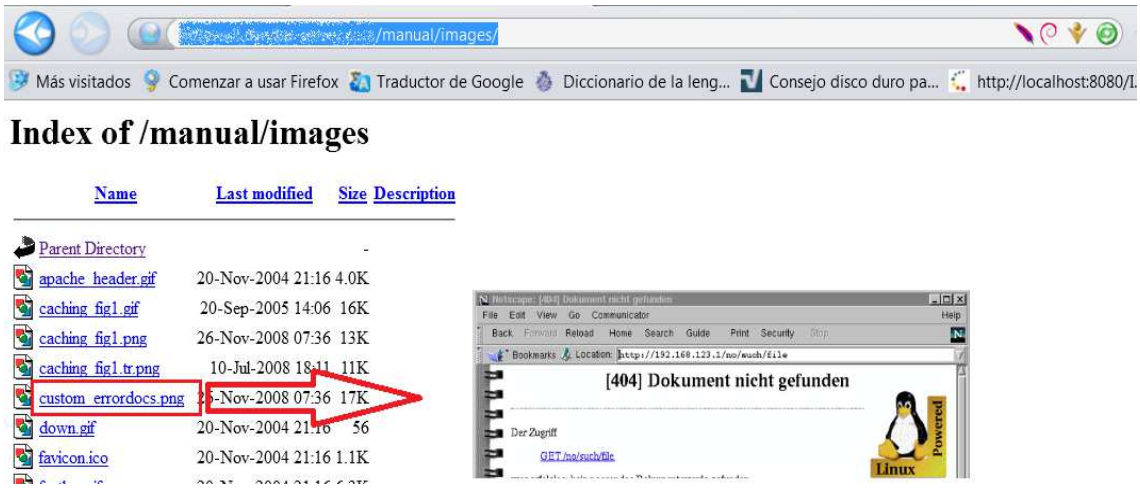


Figura 49: acceso indebido al directorio de imágenes de Apache y a los recursos

Obviamente, esto no debería suceder. Tenemos acceso a múltiples rutas que no deberían ser públicas, y a los recursos que guardan. Como vemos en la imagen, al clicar sobre "**custom_errordocs.png",** se nos abre la imagen que señalamos con la flecha. Además de Nikto, ZAP también detectó otra ruta no pública durante el "spidering" lanzado en el mapeado: `/demosmelting/contenido/`. Cargamos la ruta y el navegador muestra otro índice de ficheros accesibles. Probamos uno de ellos, y el resultado es un error que nos revela más información:

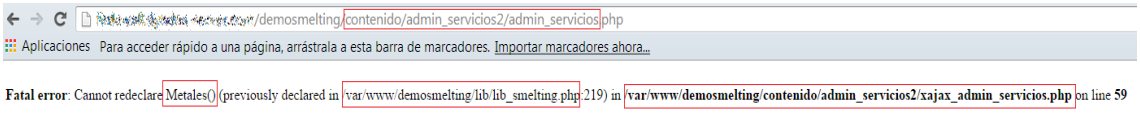


Figura 50: error al abrir el fichero admin_servicios2 del índice /contenido/

En conclusión, hemos descubierto otra vulnerabilidad de riesgo medio que habrá que documentar en el informe.

6.2.5 PHPMYADMIN

La herramienta Nikto, empleada durante la fase de mapeado, nos indicó, además de lo ya comentado, la existencia de otro directorio donde podría haber vulnerabilidades: el directorio "**/phpmyadmin/**".

PhpMyAdmin[33] es una herramienta escrita en el lenguaje PHP que se utiliza para administrar bases de datos MySQL desde el navegador. Permite crear, modificar y borrar bases de datos, así como alterar las tablas, crear tablas nuevas y ejecutar sentencias SQL, entre otras tareas.

El problema de esta aplicación es que tiene vulnerabilidades conocidas, y un buen número de exploits para atacarlas, que un hacker podría utilizar contra ella si la descubre en la web. Por este motivo, antes de nada, vamos a comprobar el aviso de Nikto, introduciendo a mano la ruta donde éste nos indica que está alojada la herramienta [46]:

[33] http://www.phpmyadmin.net/home_page/index.php

Figura 51: acceso a la consola de la herramienta phpmyadmin

Efectivamente, hemos logrado acceder a la consola de administración de phpMyAdmin. El hecho de que esté disponible desde Internet es una vulnerabilidad en sí misma, ya que un atacante que la descubra podrá hacer ataques de fuerza bruta con el objetivo de ganar acceso a la base de datos a través de la misma. En caso de que tenga éxito, tendrá control total sobre la base de datos del servidor. Por esto, este tipo de consolas de administración no deben estar disponibles desde Internet nunca; sólo deben ser accesibles desde ciertas direcciones IP o a través de redes privadas tipo VPN's.

Dado semejante descubrimiento, lo siguiente es explotarlo. Para ello, buscamos los exploits publicados más actuales en la web "Exploit Database"[34], y aplicamos los de las versiones más recientes para Linux:

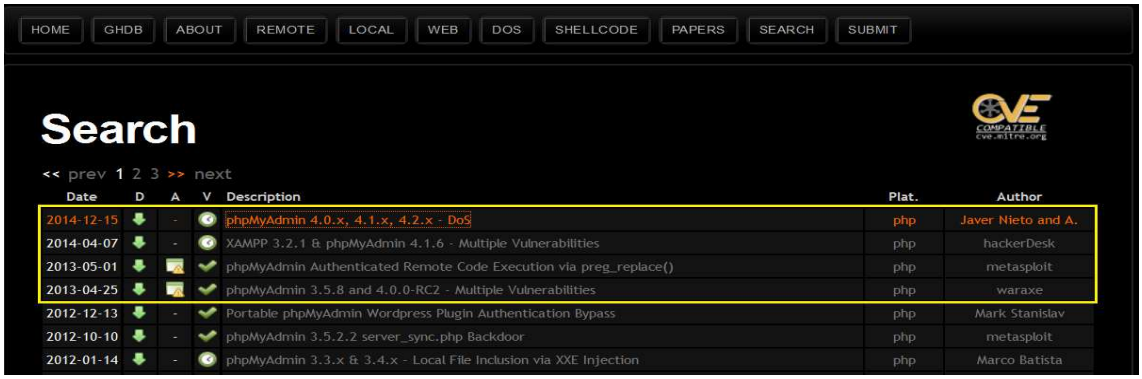


Figura 52: web exploit-db que muestra los últimos exploits contra phpmyadmin

Ninguno de los que hemos probado ha tenido éxito, debido a que la versión instalada de la herramienta no es vulnerable. El siguiente paso es realizar ataques de fuerza bruta contra la consola, para ver si logramos dar con algún usuario válido y su clave para entrar. Usamos el "fuzzer" de ZAP de nuevo. En la imagen vemos como ésta función está probando un listado de claves, que logra enviar al servidor de forma exitosa:

[34] http://www.exploit-db.com/

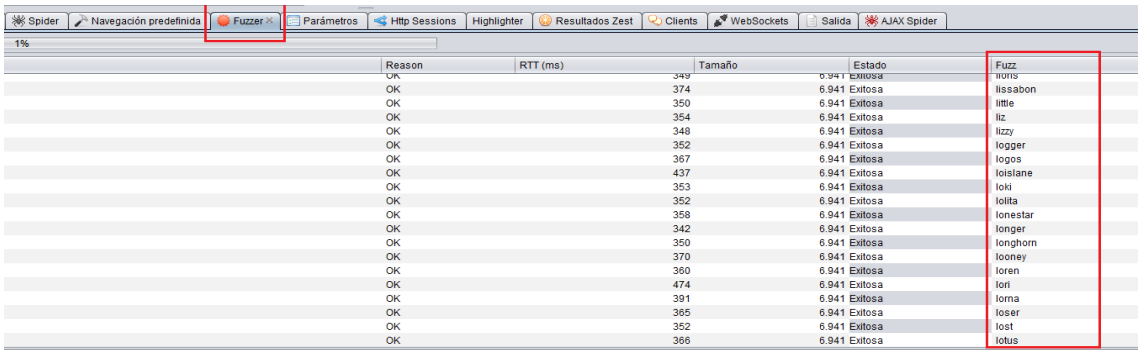


Figura 53: escaneos de fuerza bruta contra phpmyadmin

A pesar de todos los intentos, no conseguimos obtener credenciales válidas de acceso. Concluimos que la aplicación no es vulnerable, aunque sí convendría hacerla más inaccesible a los intentos maliciosos de otros hackers, y por eso, añadiremos una recomendación en el informe a este respecto.

CAPÍTULO 7. CONCLUSIONES

Llegados a este punto, sólo nos queda valorar el trabajo realizado y compararlo con los objetivos propuestos al inicio del mismo, para ver si se han cumplido satisfactoriamente. Repasamos uno a uno los objetivos y vemos si las actividades realizadas plasman el planteamiento inicial.

OBJETIVOS GENERALES

I. Síntesis de conocimientos.

A lo largo de la carrera he estudiado a fondo las redes de ordenadores, así como los sistemas operativos y lenguajes de programación de uso más extendido hasta conseguir familiarizarme con ellos. Todos estos conocimientos me han servido para entender el funcionamiento de los sistemas y aplicaciones, lo suficientemente a fondo para desarrollar la labor de técnico de sistemas y aprender la mejor manera de asegurarlos. Asignaturas como administración de redes, bases de datos, seguridad de los sistemas, o metodología y gestión de proyectos, han resultado muy útiles a la hora de saber por dónde profundizar en los conceptos, y cómo organizar un trabajo tan amplio.

II. Aproximación práctica al ejercicio de la auditoría.

Después de estudiar las metodologías teóricas, he trasladado lo aprendido a un ejercicio práctico de auditoría web, enfocado desde el punto de vista de un profesional de la consultoría en seguridad. Para ello, me he entrevistado con un consultor, y me he puesto al día acerca del modo de trabajar en la empresa hoy día, de las consideraciones más importantes a la hora de afrontar un test de penetración para un cliente, y del conjunto de herramientas de uso más extendido hoy día que un técnico debe conocer. También me he entrevistado con un desarrollador web para saber si está al tanto de los ataques web más frecuentes, si conoce la forma de evitarlos, y para saber qué importancia da a la seguridad en su trabajo.

III. Aprendizaje y puesta en práctica de las metodologías más conocidas.

En línea con el objetivo anterior, el estudio de las metodologías más conocidas me ha llevado a ampliar mis conocimientos en el campo de la auditoría web, los sistemas operativos, y las redes, y a organizar mejor las fases de mi trabajo. También, a entender las diferencias entre distintos tipos de test, a saber documentar hallazgos o construir un informe con los aspectos más importantes. Con el uso de herramientas que desconocía he aprendido, por ejemplo, a distinguir entre un balanceador de carga, un servidor virtual con diferentes nombres para diferentes servicios, o un WAF; a analizar peticiones y respuestas de un proxy, capturarlas, cambiarlas o decodificarlas con ZAP; a aprovechar el potencial de los buscadores delimitando mejor las búsquedas con sus directivas y operadores; etc.

OBJETIVOS ESPECÍFICOS

IV. Mejora del proceso de auditoría desarrollado en las metodologías.

Las metodologías estudiadas son muy amplias y abarcan todo el conjunto de posibilidades de la auditoría desde un punto de vista general. Además, listan un conjunto de pruebas para diferentes categorías de ataques, diferentes programas afectados, diferentes lenguajes, etc. En este proyecto he intentado concretar un proceso a seguir por fases ordenadas, por las que un auditor pueda avanzar en un orden algo más riguroso sin perderse entre tanta cantidad de información. Se ha intentado que cada fase tenga un límite de profundidad menor que la siguiente y mayor que la anterior. Todas las fases son prácticas. Reúnen un grupo de actividades con un fin común: reconocer el terreno, trazar un mapa, detectar puntos peligrosos y verificar esos puntos. De este modo, las fases pretenden dividir las tareas de la auditoría en escalones separados, pero muy relacionados entre sí, para ir ampliando la visión del auditor paso a paso, y que el proceso sea más eficiente en su conjunto.

V. Concienciación acerca de las ventajas del desarrollo de software seguro.

El trabajo pretende mostrar que los riesgos son reales, que incluso alguien con poca experiencia, pero mucho interés y tiempo, puede llegar a hacer estragos en una aplicación mal construida. Mostrar estos riesgos, y lo que se puede llegar a obtener de la explotación de un fallo, ayudará a concienciar al lector al respecto de tomarse en serio el desarrollo de software seguro, así como el uso de funciones seguras.

VI. Realización con herramientas de software libre.

Hoy día existen muchas páginas en Internet sobre hacking que enseñan métodos de intrusión, e incluso canales de Youtube con videos tutoriales que enseñan técnicas de hacking, o el uso de las distintas herramientas que se pueden utilizar. Por este motivo, cada año aumentan los ataques, ya que hacer hacking está al alcance de más personas, y se puede aprender con menos esfuerzos. Para demostrarlo, he realizado toda la auditoría con herramientas de software libre encontradas en estas páginas dedicadas al hacking, desarrolladas por los propietarios de los blogs donde se pueden descargar.

VII. Guía de aprendizaje.

Todo el trabajo realizado, tanto la lectura de documentación como los ejercicios prácticos, me han servido para aprender de forma más práctica sobre la seguridad de los sistemas, y más en concreto, de las aplicaciones web. En este documento he intentado plasmar, sobre todo, todas las horas dedicadas a las actividades prácticas que he ido realizando, con la idea de sintetizar un método que un estudiante de la misma disciplina pueda aprovechar también como guía de aprendizaje.

GLOSARIO

Hacking (significado tradicional): es la manipulación de la tecnología con la intención de realizar algo para lo que ésta no ha sido diseñada.

Hacker de sombrero negro (Black hat): los atacantes que comprometen sistemas o redes sin tener permiso expreso por parte del propietario.

Hacking ético: es el proceso de encontrar fallos de seguridad, mediante el empleo de técnicas de ataque conocidas, pero disponiendo del permiso expreso del propietario de los sistemas que se van a atacar. El objetivo del proceso es el de ayudar a mejorar el nivel de seguridad de los sistemas o aplicaciones objetivo. A las personas que realizan proyectos de hacking ético se las suele llamar hackers de sombrero blanco (White hat).

Vulnerabilidad: es un fallo en nuestro entorno que un atacante puede emplear para provocar daños. Por ejemplo puede haber vulnerabilidades a nivel de arquitectura de red, procesos de negocio, software empleado y configuraciones de los sistemas.

Amenaza: es el agente que puede causar daño a la organización. Los ejemplos más usuales son el crimen organizado, compañías de espionaje y empleados enfadados que atacan a su propia empresa. También caben en esta categoría los gusanos y otros tipos de malware, porque pueden provocar daños en la organización, aunque no estén dirigidos por manos humanas.

Riesgo: en el caso de que se superponga una amenaza y una vulnerabilidad, es decir, tendremos un riesgo si tenemos una vulnerabilidad en nuestros sistemas que una amenaza puede atacar.

Test de intrusión/penetración: es un tipo de prueba que se focaliza en el compromiso de los sistemas objetivo, con la intención de acceder a la información que estos almacenan.

Auditoría de seguridad: es un tipo de prueba que analiza los sistemas o aplicaciones objetivos desde la perspectiva del cumplimiento de estándares. Para realizarlos se emplean unas listas de aspectos interesantes a revisar.

Objetivo: es el sistema o aplicación que se analiza durante el proceso de auditoría o test de intrusión.

Escaneo de puertos: es la acción de analizar el estado de los puertos de una máquina con la intención de ver que servicios tiene disponibles, las versiones de los mismos y el sistema operativo utilizado. Es una técnica empleada durante los test de intrusión con la intención de encontrar servicios vulnerables, para poder explotarlos.

Exploit: es el código o la técnica que una amenaza puede emplear para aprovecharse de una vulnerabilidad. Puede ser un programa que genera paquetes que causan un desbordamiento de un búfer en el software del objetivo o puede ser emplear habilidades de ingeniería social para conseguir que la persona objetivo nos diga sus credenciales de acceso.

Spidering: es una técnica empleada para identificar los recursos disponibles en una página web. Consiste en visitar la URL solicitada y analizar el contenido de la misma en busca de más enlaces. Se emplea un proceso iterativo que se repite visitando los nuevos enlaces encontrados hasta que se han encontrado todos los recursos enlazados disponibles en la aplicación web.

Fuzzing: es un proceso de prueba del software que puede ser automático o semi-automático. Durante este proceso se envían multitud de valores a todas las entradas de un programa de software. Los valores enviados consisten en datos válidos, datos inválidos, datos aleatorios, etc. Durante estas pruebas se monitoriza el comportamiento de la aplicación objetivo, ya que esta puede mostrar errores o puede mostrar comportamientos inesperados.

Backdoor: la traducción literal es puerta trasera y consiste en un programa que puede ser instalado en un sistema o aplicación que permita el acceso a un atacante de forma inadvertida. Normalmente las puertas traseras se emplean para acceder a los sistemas saltándose todas las medidas de seguridad de los mismos y suelen ser empleados por atacantes malintencionados que quieren asegurarse de que podrán acceder a los sistemas comprometidos en el futuro.

DNS (Domain Name System): es el sistema de resolución de nombres que permite poder acceder a sistemas por nombres en lugar de utilizando sus direcciones IP. Por ejemplo, www.yahoo.com

WAF (Web Aplication Firewall): es un firewall específico para proteger aplicaciones web. Detecta patrones de ataques conocidos y evita que lleguen al servidor web.

CPD: Centro de procesamiento de datos, también conocido en inglés como “Data Center”; es una sala refrigerada, donde se instalan los servidores y otros recursos electrónicos de una organización. Estos recursos se utilizan para realizar el procesamiento de la información de la organización, o para su almacenamiento y mantenimiento.

Malware: software malicioso de todo tipo. Engloba: virus, gusanos, troyanos, espías, spam, y otros.

Log: cuaderno de bitácora; registro de sucesos que se mantiene a lo largo del tiempo.

BIBLIOGRAFÍA

A continuación mostramos un conjunto de fuentes consultadas para la realización de los diferentes capítulos de este trabajo:

[1] Fundación Wikipedia Inc. Artículo: "Auditoría informática". Última fecha de modificación: 5 de agosto de 2014. [Última fecha de consulta: 10 de octubre de 2014]. http://es.wikipedia.org/wiki/Auditor%C3%ADa_inform%C3%A1tica

[2] Fundación Wikipedia Inc. Artículo: "Historia de Internet", cronología. Última fecha de modificación: 18 de noviembre 2014. http://es.wikipedia.org/wiki/Historia_de_Internet

[3] Gits Informática – Ciber-seguridad informática y delitos telemáticos. Artículo: "¿Qué es un delito informático, ciberdelito o cibercrimen?" [Fecha de consulta: 6 de octubre de 2014]. http://www.gitsinformatica.com/legislacion.html

[4] Gits Informática – Ciber-seguridad informática y delitos telemáticos. Artículo: "Hacking, hacking ético, hacktivismo y seguridad empresarial". [Fecha de consulta: 6 de octubre de 2014]. http://www.gitsinformatica.com/hackers.html

[5] OSSTMM 3.0 – Apartado: "Process controls". Capítulo 1: páginas de 20 a 30. http://www.isecom.org/mirror/OSSTMM.3.pdf

[6] OSSTMM 3.0 – "The open source security testing methodology manual". Capítulos del 7 al 11: páginas desde la 105 a la 185. http://www.isecom.org/mirror/OSSTMM.3.pdf, http://scadahacker.com/library/Documents/Assessment_Guidance/OSSTMM-3.0.pdf

[7] Artículo UNAD (Universidad Nacional abierta a distancia), "Lección 27: resumen OSSTMM". http://datateca.unad.edu.co/contenidos/233016/EXE_SAM/leccin_27_osstmm.html

[8] Fundación OWASP, página official: "Open web application security project". Guías de pruebas, versiones 3.0 y 4.0: https://www.owasp.org/images/8/80/Gu%C3%ADa_de_pruebas_de_OWASP_ver_3.0.pdf https://www.owasp.org/images/5/52/OWASP_Testing_Guide_v4.pdf

[9] Fundación OWASP, página official: "OWASP top ten Project". Versión 2013: https://www.owasp.org/index.php/Top_10_2013-Top_10

[10] Artículo UNAD (Universidad Nacional abierta a distancia), "Lección 28: resumen OWASP": http://datateca.unad.edu.co/contenidos/233016/EXE_SAM/leccin_28_owasp.html

[11] Blog "Seguridad informática hoy", Artículo: Metodologías y herramientas de Ethical hacking [19 febrero 2013] http://seguridadinformaticahoy.blogspot.com.es/2013/02/metodologias-y-herramientas-de-ethical.html

[12] Página web oficial de A2Secure – compañía de seguridad. Sección "Auditorías". Apartado "Test de intrusión": http://www.a2secure.com/auditorias/test-de-intrusion

[13] Blog "Seguridad Informática", Artículo "Metodología de test de intrusión ISSAF" [14 abril 2009] http://insecuredata.blogspot.com.es/2009/04/metodologia-de-test-de-intrusion-issaf.html

[14] ISSAF – "Information systems security assessment framework". Artículo UNAD (Universidad Nacional abierta a distancia), "Lección 29: resumen ISSAF": http://datateca.unad.edu.co/contenidos/233016/EXE_SAM/leccin_29_issaf.html

[15] UOC (Universidad Oberta de Catalunya). Asignatura "seguridad en redes de computadores". Asignatura "metodología y gestión de proyectos informáticos – análisis de riesgos, planificación, y definición del alcance de un proyecto". http://www.uoc.edu/portal/ca/index.html
[16] Fundación Wikipedia Inc. Artículo: "Herramienta Whois": http://es.wikipedia.org/wiki/WHOIS
[17] "Herramienta Nslookup": http://es.wikipedia.org/wiki/Nslookup
[18] Web norfiPC — Artículo: "Como usar el comando NSLOOKUP en Windows, ejemplos prácticos": http://norfipc.com/redes/como-usar-comando-nslookup-windows.html
[19] Web Ring of Saturn Internetworking — Herramientas basadas en web: http://networking.ringofsaturn.com/
[20] Blog SolucionesC2.com — Artículo: "Qué es Robots.txt y para qué sirve" [29 de abril 2013]: http://blog.solucionesc2.com/que-es-el-robots-txt-y-para-que-sirve-c2-seo
[21] Blog Exploit Database — Sección "Google Hacking-Database" [Última actualización 9 noviembre 2014]: http://www.exploit-db.com/google-dorks/
[22] Web del ministerio de educación, cultura y deporte del gobierno de España — "Observatorio tecnológico", sección "Internet — recursos on-line", artículo "búsquedas avanzadas en Google". Directivas y operadores de Google para Google hacking: http://recursostic.educacion.es/observatorio/web/eu/internet/recursos-online/1004-busquedas-avanzadas-en-google
[23] Web Googleguide.com — Apartado: "Using search operators - Query input I - Search operators": http://www.googleguide.com/advanced_operators_reference.html
[24] Web Youtube.com – Canal "Hacking ético web" – video: "Google Hacking Database (GHDB): http://www.youtube.com/watch?v=7RMfNZIloCs
[25] Blog "Ha.ckers.org" – Sección "Fierce" – Herramienta "Fierce domain scan": http://ha.ckers.org/fierce/
[26] Blog Dragonjar.org – Sección "Herramientas de seguridad" – Artículo: "CeWL — Generador de diccionarios personalizados" [fecha 07 enero 2010]: http://www.dragonjar.org/generador-diccionarios-passwords.xhtml
[27] Web Insecure.org – Nmap security scanner: www.insecure.org
[28] Web Taddong.com: seguridad en profundidad [2010 - 2013]. Sección "Lab: herramientas, TLSSLed v1.3". Notas sobre análisis de SSL con TLSSled:
http://www.taddong.com/es/lab.html
http://blog.taddong.com/2011/05/tlssled-v10.html
http://blog.taddong.com/2013/02/tlssled-v13.html
[29] Proyecto de software libre Wafw00f [fecha 8 noviembre 2014]: https://github.com/sandrogauci/wafw00f
[30] Herramienta Halberd. Manual "Halberd user's guide". Autor Juan M. Bello Rivas [fecha 14 agosto 2010]:
http://halberd.superadditive.com/
http://halberd.superadditive.com/doc/manual/
[31] Blog "Gurú de la informática" desde 2005. Artículo: "Auditar la seguridad de la configuración del balanceo de carga de servidores web" [fecha 11 enero 2010]: http://www.gurudelainformatica.es/2010/01/auditar-la-seguridad-de-la.html
[32] Web CIRT.net "Suspicion breeds confidence". Sección "Nikto":
https://www.cirt.net/
https://www.cirt.net/Nikto2
https://cirt.net/nikto2-docs/
[33] Fundación OWASP. Sección "ZAP proxy" [actualizado 2013]: https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project

[34] Web Youtube.com. Canal "Owasp ZAP tutorial videos":
https://www.youtube.com/playlist?list=PLEBitBW-Hlsv8cEIUntAO8st2UGhmrjUB

[35] Web blog thehackerway.com: "Seguridad en sistemas, y técnicas de hacking". Artículo Web hacking – arquitecturas web vulnerables – parte XVIII. Sección "Métodos HTTP peligrosos" [fecha 17 de enero del 2013]:
http://thehackerway.com/2013/01/17/web-hacking-arquitecturas-web-vulnerables-parte-xviii/

[36] Web Eicar "European Institute for Computer Anti-virus Research" [1998 - 2014]:
http://www.eicar.org/86-0-Intended-use.html

[37] Tutoriales para Sqlmap:
a) Web Youtube.com. Video "SQLMAP Tutorial [SQL Injection] Kali Linux". Canal "ProScout97". Publicado [29 abril 2014]. https://www.youtube.com/watch?v=y4nMgoY5fpY
b) Blog "elhacker.net". Artículo: "Tutorial – Manual SQLmap: ataques SQLi – Inyección SQL". Autor de la publicación: "el-brujo". Fecha [18 junio de 2014].
http://blog.elhacker.net/2014/06/sqlmap-automatizando-ataques-sqli-injection.html

[38] Web de php.net – Documentación – Manual de php > Seguridad de bases de datos. Artículo "Inyección de SQL". The php group [2001 – 2014].
http://php.net/manual/es/security.database.sql-injection.php

[39] Web "Stackoverflow". Sección "Questions". Artículo "How can I prevent SQL-Injection in PHP". Autor: Andrew G. Johnson. Fecha [14 mayo 2014].
http://stackoverflow.com/questions/60174/how-can-i-prevent-sql-injection-in-php

[40] Fundación OWASP. Página: XSS Filter Evasion Cheat Sheet. Última actualización [13 octubre 2014]
https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet

[41] Instituto INFOSEC – Artículo "How to prevent cross-site scripting attacks" [10 octubre 2013]
http://resources.infosecinstitute.com/how-to-prevent-cross-site-scripting-attacks/

[42] CWE: "Common Weakness Enumeration". Cwe List – Individual dictionary definition (2.8). Artículo: "CWE-79: Improper neutralization of input during web page generation ('XSS'). Última actualización [30 julio 2014].
http://cwe.mitre.org/data/definitions/79.html

[43] VERACODE – Tutorial XSS: "Learn about XSS vulnerabilities, injections and how to prevent attacks". Autor: Neil DuPaul.
http://www.veracode.com/security/xss

[44] CWE: "Common Weakness Enumeration". Cwe List – Individual dictionary definition (2.8). Artículo: "CWE-548: Information exposure through directory listing. Última actualización [30 julio 2014].
http://cwe.mitre.org/data/definitions/548.html

[45] Fundación OWASP. Artículo "OWASP periodic table of vulnerabilities – Directory Indexing". Última modificación [15 mayo 2013].
https://www.owasp.org/index.php/OWASP_Periodic_Table_of_Vulnerabilities_-_Directory_Indexing

[46] CVE Details – The ultimate security vulnerability datasource. Sección "Phpmyadmin: Security Vulnerabilities".
http://www.cvedetails.com/vulnerability-list/vendor_id-784/Phpmyadmin.html

ANEXO 1. ANÁLISIS DE RIESGOS

Algunas de las situaciones de malas prácticas, que podrían causar riesgos a largo plazo, son las siguientes:

Riesgo R01	Plazos demasiado cortos
Causa	El cliente o jefe de proyecto otorga un plazo de tiempo demasiado corto para la realización de la tarea técnica.
Descripción	La estimación de tiempo para auditar una aplicación no es adecuada.
Consecuencia	Auditoría incompleta. Por tanto, de escasa utilidad.
Probabilidad	Media
Impacto	Alto
Nivel	Alto
Acción de Contingencia	Otorgar plazos adecuados. La estimación debe realizarse teniendo en cuenta la opinión del auditor o la participación del mismo.

Figura 54: Riesgo R01

Riesgo R02	Técnico poco experimentado
Causa	Se encarga la tarea a un técnico sin experiencia previa, lo que da como resultado una búsqueda infructuosa de los fallos.
Descripción	La entidad prefiere contratar a un técnico en prácticas sin experiencia, en lugar de uno experimentado pero más costoso.
Consecuencia	Falsos negativos. Resultados poco fiables.
Probabilidad	Media
Impacto	Alto
Nivel	Alto
Acción de Contingencia	Designar a técnicos cualificados para la revisión de aplicaciones críticas. Dar formación adecuada para técnicos poco experimentados de la mano de personal con más experiencia. Designar servidores de pruebas para entrenamiento inicial.

Figura 55: Riesgo R02

Riesgo R03	Falta de periodicidad
Causa	El ataque se produce antes de la finalización de la auditoría, o de la reparación o parcheo de la vulnerabilidad.
Descripción	Las auditorías deben hacerse de forma periódica, ya que con el tiempo, pueden descubrirse nuevos fallos de seguridad. Si una aplicación se pone en producción sin haberla revisado, puede darse el caso de que la aplicación sea atacada antes de haber pasado una auditoría.
Consecuencia	Proceso de auditoría fallido. Se ha violado la condición de confidencialidad, de integridad, o de disponibilidad del sistema.
Probabilidad	Alta
Impacto	Alto
Nivel	Alto
Acción de Contingencia	Implantar la seguridad como proceso. Incluir auditorías periódicas en el ciclo de vida del desarrollo de software. Auditar la aplicación antes de su puesta en producción.

Figura 56: Riesgo R03

Riesgo R04	Falta de implicación del cliente
Causa	El propietario de la web no hace caso de las recomendaciones recibidas en el informe del auditor, fruto de una auditoría, por darles poca relevancia.
Descripción	Si el cliente no entiende los riesgos y no aplica las recomendaciones, por no creerlas necesarias, la seguridad de su aplicación seguirá estando expuesta.
Consecuencia	Auditoría inválida o inservible.
Probabilidad	Media
Impacto	Alto
Nivel	Alto
Acción de Contingencia	Implicar al cliente en el proceso. Llevar a cabo las modificaciones recomendadas.

Figura 57: Riesgo R04

Riesgo R05	Incumplimiento del acuerdo de confidencialidad
Causa	La falta de formación o profesionalidad del auditor puede llevarle a no cumplir el acuerdo de no divulgación de información.
Descripción	El técnico es descuidado y divulga las vulnerabilidades encontradas, propiciando ataques de terceros contra sus clientes.
Consecuencia	Seguridad comprometida. Reputación de la entidad auditora perdida.
Probabilidad	Baja
Impacto	Alto
Nivel	Alto
Acción de Contingencia	Verificar que los acuerdos se cumplen por todas las partes implicadas. Enseñar al personal el código de conducta apropiado y las buenas prácticas de la profesión.

Figura 58: Riesgo R05

ANEXO 2. PLANIFICACIÓN TEMPORAL DETALLADA

1.1. CALENDARIO DE REALIZACIÓN: HITOS

La fecha fijada para el inicio del proyecto es el día 17 de septiembre del 2014, y la fecha de finalización, el 9 de enero del 2015. Por tanto, tenemos un total de 115 días, contando fines de semana y festivos. Sin embargo, el tiempo disponible no es total, por lo que estimaremos en principio una jornada de trabajo de 3 horas los días entre semana, y entre 5 y 6 los fines de semana y festivos. Seguidamente, dividiremos el total de días asignados entre cuatro entregas parciales.

Acerca de la división de los días disponibles entre las tablas: el campo "nº de días" de cada tabla no se refiere a que la suma de los días de todas sus casillas dé igual al total de días asignados a esa entrega. Habrá muchas actividades que se harán en paralelo, o que se harán a lo largo de todo el proyecto y no en un momento determinado. Por ejemplo, la lectura de documentación o recopilación de resultados, son actividades que se irán realizando conforme se avance el proyecto, por eso el nº de días asignados es el total del periodo de entrega, lo que no quiere decir que durante esos días no se realicen a la vez otras actividades. Además, el proceso será cíclico, y puede ocurrir que en un estadio avanzado debamos volver a ejecutar tareas de un estadio anterior, por ejemplo, durante la explotación podría darse el caso de tener que volver a hacer alguna tarea de la fase de reconocimiento o descubrimiento.

1.1.1. PLAN DE TRABAJO

La tarea principal de esta fase es la realización del plan de trabajo, una vez seleccionado un proyecto. Para ello contamos con un total de 20 días que asignaremos a las siguientes tareas:

PAC1	INICIO	FIN	Nº DE DÍAS
Selección de proyecto	17/09/2014	19/09/2014	2
Descarga de material de apoyo del aula de la UOC	19/09/2014	20/09/2014	1
Lectura del material y del plan docente	20/09/2014	21/09/2014	1
Resolución de dudas	21/09/2014	23/09/2014	2
Elaboración del plan de trabajo completo	23/09/2014	06/10/2014	14
Lectura/visionado de documentación	17/09/2014	06/10/2014	20

Figura 59: Tareas que componen la primera entrega

1.1.2. SEGUNDA ENTREGA PARCIAL

Las tareas a desarrollar en esta fase serán, principalmente, las que compondrán las fases de reconocimiento y mapeado de la auditoría. Además, se empezará a hacer la toma de evidencias, a componer el informe de resultados y la memoria, y se continuará con la tarea, ya empezada, de lecturas recomendadas. Para todo esto contaremos con 35 días, que asignaremos como sigue:

PAC2	INICIO	FIN	Nº DE DÍAS
Comienzo del Reconocimiento	**06/10/2014**	**16/10/2014**	**10**
Definición del alcance	06/10/2014	08/10/2014	2
Búsquedas en registros de internet	08/10/2014	10/10/2014	2
Consultas en páginas públicas	10/10/2014	12/10/2014	2
Consultas con motores de búsqueda	12/10/2014	14/10/2014	2

Búsqueda de sub-dominios	14/10/2014	15/10/2014	1
Elaboración de diccionarios	15/10/2014	16/10/2014	1
Comienzo del mapeado	**17/10/2014**	**07/11/2014**	**20**
Escaneo de puertos	17/10/2014	19/10/2014	2
Búsqueda de versión de SO y otras aplicaciones	19/10/2014	21/10/2014	2
Análisis SSL	21/10/2014	25/10/2014	4
Análisis de balanceo de carga y WAF	25/10/2014	29/10/2014	4
Análisis de la configuración del software	29/10/2014	02/11/2014	4
Spidering	02/11/2014	06/11/2014	4
Toma de evidencias/informe	**06/10/2014**	**06/11/2014**	**30**
Elaboración de la PAC2/memoria	**08/11/2014**	**10/11/2014**	**35**
Lectura/visionado de documentación	**06/10/2014**	**10/11/2014**	**35**

Figura 60: Tareas que componen la segunda entrega

1.1.3. TERCERA ENTREGA PARCIAL

Las tareas a desarrollar en esta fase serán, principalmente, las que compondrán las fases de descubrimiento y explotación de la auditoría. Además, se continuará con la toma de evidencias, el informe de resultados, y la composición de la memoria. Para ello contaremos con un total de 39 días que asignaremos como sigue:

PAC3	INICIO	FIN	Nº DE DÍAS
Fase de descubrimiento	**10/11/2014**	**05/12/2014**	**25**
Escaneos automáticos web	10/11/2014	23/11/2014	13
Fuzzing de los parámetros de la app.	24/11/2014	05/12/2014	12
Fase de explotación	**06/12/2014**	**19/12/2014**	**14**
Lanzamientos de diferentes ataques	06/12/2014	19/12/2014	14
Toma de evidencias/informe	**10/11/2014**	**15/12/2014**	**35**
Elaboración de la PAC3/memoria	**16/12/2014**	**19/12/2014**	**39**

Figura 61: Tareas que componen la tercera entrega

1.1.4. ENTREGA FINAL: MEMORIA Y PRESENTACIÓN

En esta última fase concluiremos con la memoria final del proyecto y el informe resultado de la auditoría del caso práctico, ambas tareas en paralelo, y además realizaremos la presentación virtual. El periodo completo es de 21 días, que se dividirán como sigue:

PAC4	INICIO	FIN	Nº DE DÍAS
Conclusión de la memoria final	19/12/2014	29/12/2014	10
Conclusión del informe de auditoría	30/12/2014	02/01/2015	10
Realización de la presentación	03/01/2015	09/01/2015	11

Figura 62: Tareas que componen la cuarta entrega

1.2. RESUMEN DE FECHAS CLAVE

Para finalizar, resumimos en una tabla las fechas de inicio y fin de las entregas parciales o hitos más importantes de todo el proyecto:

ENTREGAS PARCIALES	INICIO	FIN	Nº DE DÍAS
PAC1: plan de trabajo	17/09/2014	06/10/2014	20
PAC2: segunda entrega parcial	06/10/2014	10/11/2014	35
PAC3: tercera entrega parcial	10/11/2014	19/12/2014	39
PAC4: memoria y producto resultado	19/12/2014	02/01/2015	14
Presentación virtual	02/01/2015	09/01/2015	7

Figura 63: Tabla de fechas clave

ANEXO 3. PERFIL DEL AUDITOR: CAPACIDADES Y HABILIDADES

A continuación, presentamos un listado de las habilidades más útiles del rol de auditor, o las capacidades más deseables para el desempeño de su trabajo:

3.1. PENSAMIENTO CREATIVO Y MENTE ANALÍTICA

La labor de auditor consiste en revisar el trabajo realizado por otros técnicos, y encontrar los puntos de fallo y errores que estas personas no han sabido hallar o evitar. Esta labor se desarrolla sobre disciplinas de todo tipo, por ejemplo, si lo que se audita es un código fuente, este código puede estar escrito en cualquier lenguaje. Como el auditor no puede conocerlo todo, debe valerse de la creatividad para realizar su función, por un lado, y de su capacidad analítica por otro.

3.2. CARÁCTER ORGANIZADO Y METÓDICO

No es aconsejable realizar todo el trabajo técnico sin parar a tomar evidencias de cada prueba, tanto si es exitosa como si no, y dejar la labor de documentación para el final. Esto puede llevar a que el auditor olvide algún hallazgo, y su informe quede incompleto. Ser organizado y metódico, saber tomar buenas notas, o esquematizar todas las tareas pendientes en un diagrama, resultará más ventajoso que recurrir a la memoria. Por eso, es aconsejable elaborar el informe y tomar evidencias en paralelo al trabajo técnico, y ante todo, anotarlo todo siempre.

3.3. RESPONSABILIDAD Y DISCRECIÓN

Los datos que los auditores manejan son sensibles y confidenciales. No le pertenecen a él sino a la empresa que lo contrata, y suelen ser activos de mucho valor para esta. Por estas razones, el auditor debe ser muy discreto con todo aquello que pasa por sus manos, y tener cuidado de no revelar la información de sus clientes. Por otro lado tiene la responsabilidad profesional de no aprovecharse de los fallos que encuentre. Si por ejemplo, da con una tienda "on line" con un fallo en su plataforma de cobro, que hace que el producto se venda gratis, el auditor no debe aprovechar esto para su lucro personal. Por otro lado, si el cliente intenta presionar para que modifique de algún modo el resultado de la prueba, el auditor tiene la responsabilidad de no ceder a esas presiones.

3.4. BUENAS DOTES COMUNICATIVAS

El auditor debe saber comunicar su trabajo tanto de forma oral como escrita. Una característica importante de su trabajo, es que debe ser repetible. Esto significa que otras personas, aparte de él, deben ser capaces de reproducir los problemas encontrados y documentados en el informe de resultados. Además, deben poder hacerlo sin tener el nivel de conocimientos técnicos que tiene el auditor, por lo que cada hallazgo debe estar explicado de modo que sea comprensible para todo destinatario.

3.5. CONOCIMIENTO DE INGLÉS

Aparte de poseer conocimientos elevados en las tecnologías de la información, el conocimiento de idiomas, y en concreto de inglés, será vital para el auditor. Los clientes serán empresas de todas partes del mundo, y la comunicación con ellos se desarrollará en inglés. La redacción de los informes se pedirá en inglés en muchos casos y, por supuesto, la mayoría de la documentación sobre vulnerabilidades, las fuentes para indicar recomendaciones, los cursos, metodologías, conferencias, material de estudio para certificaciones, apuntes y videos explicativos sobre herramientas, etc, casi todo estará en inglés. Así que, un buen nivel de inglés será casi imprescindible.

ANEXO 4. BUENAS PRÁCTICAS Y HÁBITOS DEL AUDITOR

Para desempeñar la labor de auditoría, hay una serie de buenas prácticas que conviene conocer y adoptar entre los hábitos de trabajo diarios del auditor:

4.1. COMUNICACIÓN FLUIDA CON EL CLIENTE

Antes de realizar alguna prueba de explotación peligrosa, que pueda saturar la máquina destino o causar algún otro efecto incómodo, conviene descolgar el teléfono para avisar al cliente. Éste se interesará por saber qué clase de pruebas son, y qué efectos podrían causar en su servidor. Quedará mucho más tranquilo y conforme si puede oír la explicación de boca del propio técnico. Se debe acordar con él un momento propicio, en que ambas partes estén atentas y se pueda reiniciar el servidor en caso de sobrecarga al menor tiempo posible.

4.2. TOMA DE EVIDENCIAS

Las evidencias que se registren deben ser explicativas, comprensibles por sí mismas, que muestren la vulnerabilidad encontrada y ayude a reproducirla de nuevo por otras personas. Además, es aconsejable tomarlas en el momento mismo en que se hace la prueba, para evitar tener que repetir trabajo. Ésta es una tarea constante que se realiza a lo largo de toda la auditoría, y forma parte de todas sus fases.

4.3. INTERCAMBIO DE DATOS

Todo intercambio de datos confidenciales entre cliente y auditor debe realizarse en modo cifrado. Esto incluye el envío de informes, de documentación, o de credenciales. Hay programas para crear pares de claves, pública y privada, con las que poder cifrar y encriptar todo archivo que enviemos por la red. Así, en caso de ser interceptado, el fichero no podrá ser leído.

4.4. COPIAS DE SEGURIDAD

Todo ingeniero en informática debe acostumbrarse a realizar copias periódicas de su trabajo. En el caso del auditor, con más razón si cabe. Las copias deben hacerse en un dispositivo diferente del que almacena el trabajo original, a ser posible, en un dispositivo externo que pueda guardarse en una habitación distinta. Los datos confidenciales se deben guardar en medios cifrados, y en ficheros cifrados. Hay que tener presente que aunque el auditor tenga permisos para visionar la información, puede que el resto de su departamento no lo tenga, ni tampoco el resto de departamentos de la empresa. Si se utilizan dispositivos externos transportables, nunca deben sacarse de la empresa sin cifrar.

4.5. REGISTRO DE NOTAS

Aunque no parezca moderno, una de las mejores herramientas de trabajo que existen es la libreta de notas. Anotar en ella un listado de cada idea que tengamos como auditores, en el mismo instante en que surge esa idea en nuestra cabeza, ayudará a que no nos dejemos ninguna tarea sin realizar ni ninguna idea sin probar por no volver a recordarla. Ir tachando cada entrada de la lista conforme la hayamos realizado, o puntuar las entradas según su prioridad, nos ayudará a organizar mejor el volumen de trabajo, el tiempo disponible, y a estimar de forma más realista los cambios que se puedan solicitar a mitad de proyecto por el cliente (cosa que suele suceder muy a menudo). Añadir la fecha a cada nota ayuda a identificar cuándo se hizo cada prueba, dato que muchas veces

pide el cliente. Otras cosas que podemos anotar serán los errores provocados, o los comportamientos extraños que detectemos.

4.6. LOGS

Como medida de prevención frente a conflictos o disconformidades con el cliente, conviene guardar "logs" de las aplicaciones que se usen durante la auditoría.

4.7. TIEMPOS ACOTADOS

Nunca sobra el tiempo que se asigna a una auditoría. Para un auditor, una buena gestión del tiempo le facilitará mucho el trabajo. Se recomienda paralelizar las tareas automáticas con las revisiones manuales, siempre que sea posible hacerlo. Esto puede suponer un ahorro importante de tiempo, sobre todo en pruebas de caja negra, donde no podemos estimar a priori la cantidad de trabajo pendiente, porque desconocemos toda la funcionalidad de la aplicación a analizar.

4.8. HERRAMIENTA PRINCIPAL ACTUALIZADA

La herramienta principal del trabajo del auditor es su ordenador. En él, se realiza toda la actividad. Por esto, se debe ser cuidadoso, y mantenerlo debidamente protegido y actualizado. Se aconseja configurar el equipo de modo que no se pueda entrar directamente al encender, sino que sólo sea posible el acceso por medio del usuario y contraseña del auditor. Si éste deja momentáneamente su puesto, debe bloquear el equipo para evitar intrusiones. En los momentos de descanso, no conviene navegar por sitios web peligrosos, piratas, que puedan contener malware. Tampoco se aconseja hacer descargas de software ilegal, sino de sitios oficiales, o conocidos. Ser cuidadosos también con las sesiones abiertas; tenerlas siempre bajo control.

4.9. MANTENERSE AL DÍA

Una de las tareas no oficiales de cualquier profesional será la lectura diaria de noticias sobre las novedades descubiertas en el sector. Cada día aparecen nuevas vulnerabilidades y se da solución a otras. Estos cambios hay que conocerlos. Conviene reservar un tiempo a la lectura de webs de referencia, a la consulta de manuales técnicos, a la asistencia a conferencias de seguridad o cursos de formación, etc. Otra buena costumbre es compartir las novedades con otros profesionales del sector, compañeros, etc. También hay que estar al día a nivel técnico y conocer con profundidad las tecnologías con las que se trabajará.

4.10. SOFTWARE LEGAL

Todo el software que use el auditor y tenga instalado en su equipo de trabajo, debe ser legal, ya sea comercial o de código abierto. Como precaución, el software nuevo que se quiera usar para un proyecto, debe ser probado previamente en un entorno de laboratorio, nunca en producción. Así se evita que los resultados inesperados afecten a un proyecto, y con ello, a un cliente.

ANEXO 5. DIRECTIVAS Y OPERADORES DE GOOGLE

Repasamos algunas directivas y operadores más útiles para labores de hacking [22]:

2.1. "SITE:" Y "CACHE:"

La directiva "site" (disponible en varios buscadores), limita los resultados de la búsqueda a un dominio concreto, o a una web concreta. Si el auditor tiene como objetivos específicos, en su definición de alcance, URLs o dominios concretos, esta directiva le resultará muy útil para focalizarse en resultados que tengan que ver sólo con su objetivo. Por ejemplo:

```
site: wikipedia.es
```

Nos da como resultado una serie de páginas pertenecientes al dominio wikipedia.es:

Figura 64: resultado de la búsqueda en Google usando la directiva "site"

La directiva "cache" [`cache:url`] es muy parecida a "site". La diferencia es que nos muestra la versión de la página buscada que Google guarda en su caché, no la versión que actualmente está publicada. Si una web ha sido eliminada, pero Google aún la guarda en caché, podríamos acceder a ella a través de esta directiva. Si la web ha sido actualizada, podemos ver las modificaciones accediendo a estas versiones anteriores.

2.2. "FILETYPE" Y "EXT"

Estas dos directivas tienen la misma función. Sirven ambas para buscar ficheros que tengan una extensión determinada. Por ejemplo, si sabemos que nuestro cliente emite sus facturas en formato .pdf, podemos limitar la búsqueda a ficheros con esa extensión. Para comprobarlo, vamos a buscar en Google páginas de Owasp que contengan archivos en pdf. Podemos hacerlo de las dos formas, aplicando la primera directiva:

```
filetype:pdf www.owasp.org
```

O bien, aplicando la segunda:

```
ext:pdf www.owasp.org
```

Lo que obtenemos en ambos casos son una serie de enlaces directos a ficheros .pdf que se encuentran guardados en la web de Owasp y que son accesibles a través de algún link. En la imagen vemos lo que nos devuelve Google al aplicar la primera directiva:

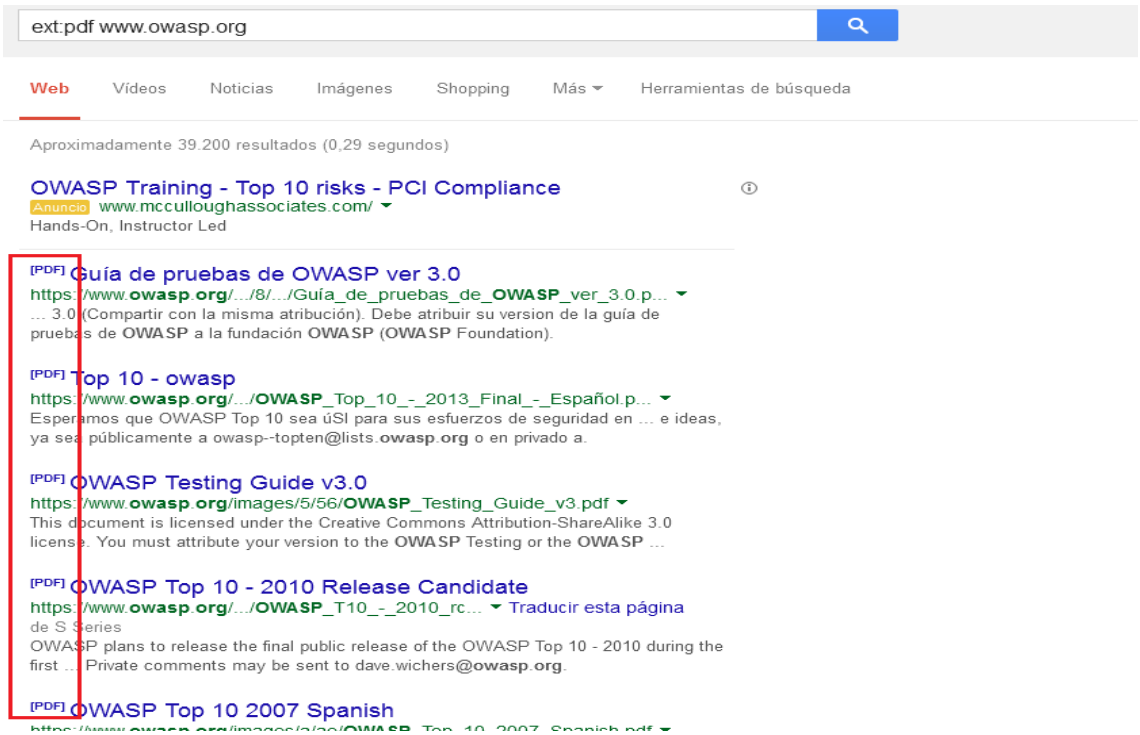


Figura 65: el recuadro rojo resalta la indicación de que cada enlace conduce a un .pdf

2.3. "INURL" Y "INTITLE"

Las dos directivas son muy similares pero no idénticas. Ambas restringirán la búsqueda a páginas que contengan una determinada palabra introducida por el usuario a continuación de los dos puntos que acompañan a la directiva [`directiva:término`].

- "`inurl:`*`término`*". Si usamos "*inurl*", Google nos devolverá sólo páginas que contengan el término concreto escrito al lado, en su URL. Por ejemplo:

```
inurl:top www.owasp.org
```

Esto nos devolverá páginas de Owasp que contengan la palabra "top" en su URL:

Figura 66: fracción de la búsqueda con inurl:top

- "`intitle:término`". Si usamos "*intitle*", Google nos devolverá páginas que contengan el termino indicado en el campo "título" de la página [`<title>término</title>`]. Por ejemplo:

```
intitle:manual www.owasp.org
```

Esto nos devolverá páginas de owasp en cuyo campo [`<title>`] se encuentre la palabra "manual".

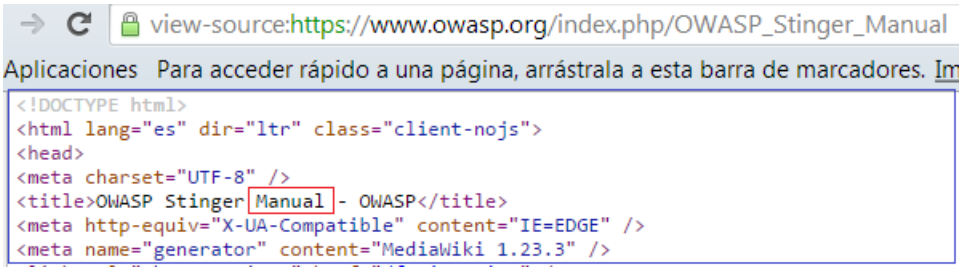


Figura 67: vemos la palabra "manual" dentro del campo <title>

Todas estas directivas, y muchas más, pueden combinarse entre sí para restringir más las búsquedas. Si un atacante no tiene un objetivo fijado, la combinación de directivas y palabras clave puede llevarle a dar con muchas páginas que son vulnerables por la red.

2.4. OPERADORES

Además de las directivas vistas, Google también ofrece operadores para acotar las búsquedas [23]. Los más conocidos son:

- **Doble comilla " "**. Encierran una frase para que sea buscada de forma literal.
- **Signo más "+"**. Al lado de una palabra, hace que ésta sea incluida en la búsqueda, a pesar de que sea habitualmente ignorada por Google.
- **Signo menos "-"**. Sirve para omitir páginas o palabras de una búsqueda. Por ejemplo, si quiero encontrar a "José Pérez" el actor, puedo excluir a "José Pérez" el futbolista añadiendo a la búsqueda: "José Pérez —futbol". Así omitiremos todos los resultados relacionados con futbol.
- **Comodín "*"**. El comodín puede sustituir cualquier palabra (una o más), en una frase encerrada entre comillas.

ANEXO 6. CONFIGURACIÓN DE ZAP PROXY

Para configurar el ZAP la primera vez que lo usemos, tras instalar el plugin de Mozilla "foxyproxy" y clicar con botón derecho sobre el icono del zorro (en la imagen, es el icono que se ve en la zona gris entre las dos barras blancas de navegación), se nos abre el menú que se ve en la imagen siguiente y que sirve para cambiar el proxy que queremos usar.

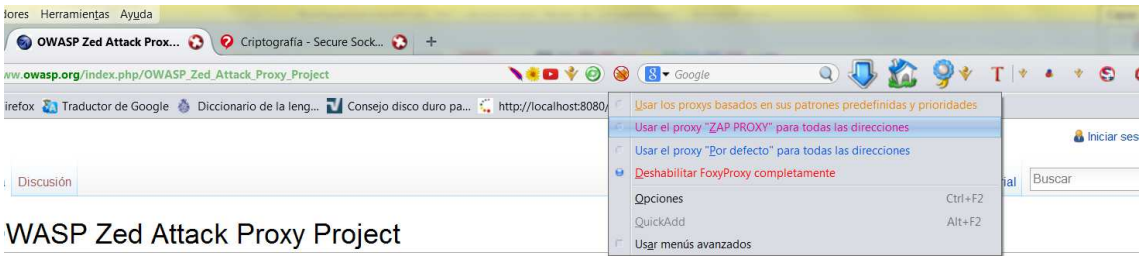


Figura 68: menú para seleccionar Zap proxy

Clicamos en la opción segunda (la de color rosa), y se nos abre otra ventana que nos lista los proxys que Firefox puede usar. Seleccionamos otra vez el de color rosa, que corresponde al de ZAP proxy. Lo vemos en la imagen siguiente:

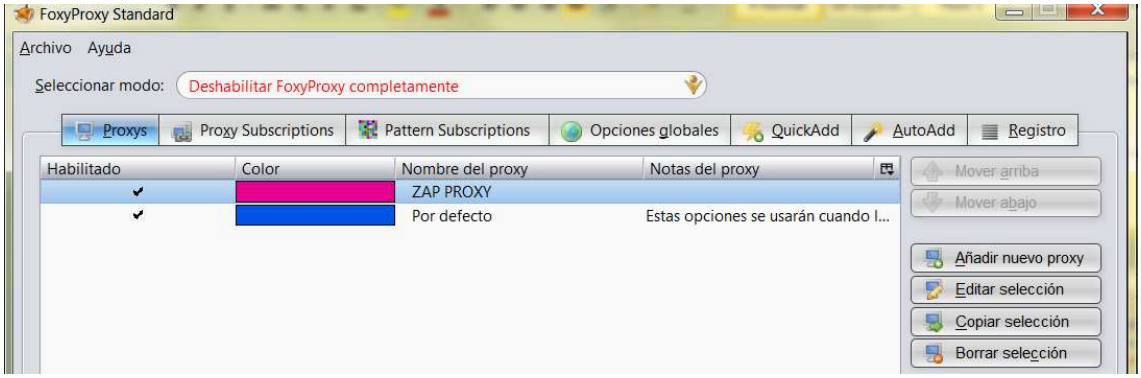


Figura 69: menú de opciones de foxyproxy

Con doble clic, se nos abre el siguiente menú, donde debemos indicar la dirección IP propia, y el puerto TCP donde está escuchando el proxy Zap:

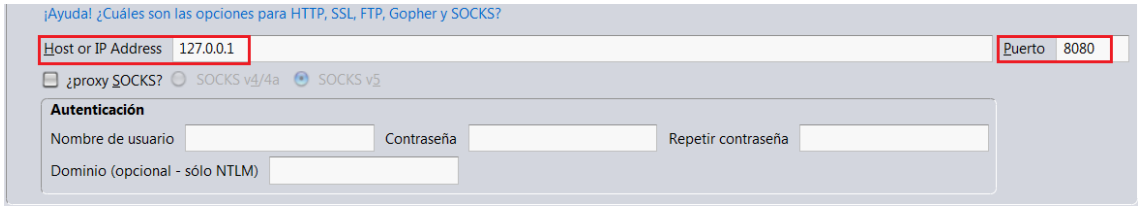


Figura 70: configuración de Firefox para capturar peticiones

Con ZAP activado, ya podemos cargar la web objetivo y empezar a navegar.