GOBIERNO DE ESPAÑA
VICEPRESIDENCIA TERCERA DEL GOBIERNO
MINISTERIO DE ASUNTOS ECONÓMICOS Y TRANSFORMACIÓN DIGITAL
SECRETARÍA DE ESTADO DE DIGITALIZACIÓN E INTELIGENCIA ARTIFICIAL

Seminario web 5 "Uso de OWASP ZAP"

Ejercicios

017 TU AYUDA EN CIBERSEGURIDAD
incibe_ INSTITUTO NACIONAL DE CIBERSEGURIDAD

ÍNDICE

ÍNDICE DE FIGURAS

1. EJERCICIO PRÁCTICO

Estudio de las comunicaciones HTTP en la aplicación bWAPP utilizando un enfoque pasivo desde una distribución Kali Linux.

El objetivo de este ejercicio es que el alumno se familiarice con la configuración de OWASP ZAP para el análisis de tráfico HTTP y que estudie las comunicaciones utilizadas con un dominio local; el host "test" servirá como apoyo; estudiaremos el tipo de parámetros enviados y recibidos, el método empleado, la localización de formularios web, etc. El alumno utilizará un enfoque pasivo, sin utilizar ninguna de las funcionalidades de ataque disponibles en OWASP ZAP.

Vamos a configurar el proxy del navegador para utilizar la dirección localhost en el puerto 8080. Asegúrese de seleccionar la opción "*Use this proxy server for all protocols*" para incluir el posible tráfico SSL en dicha configuración.

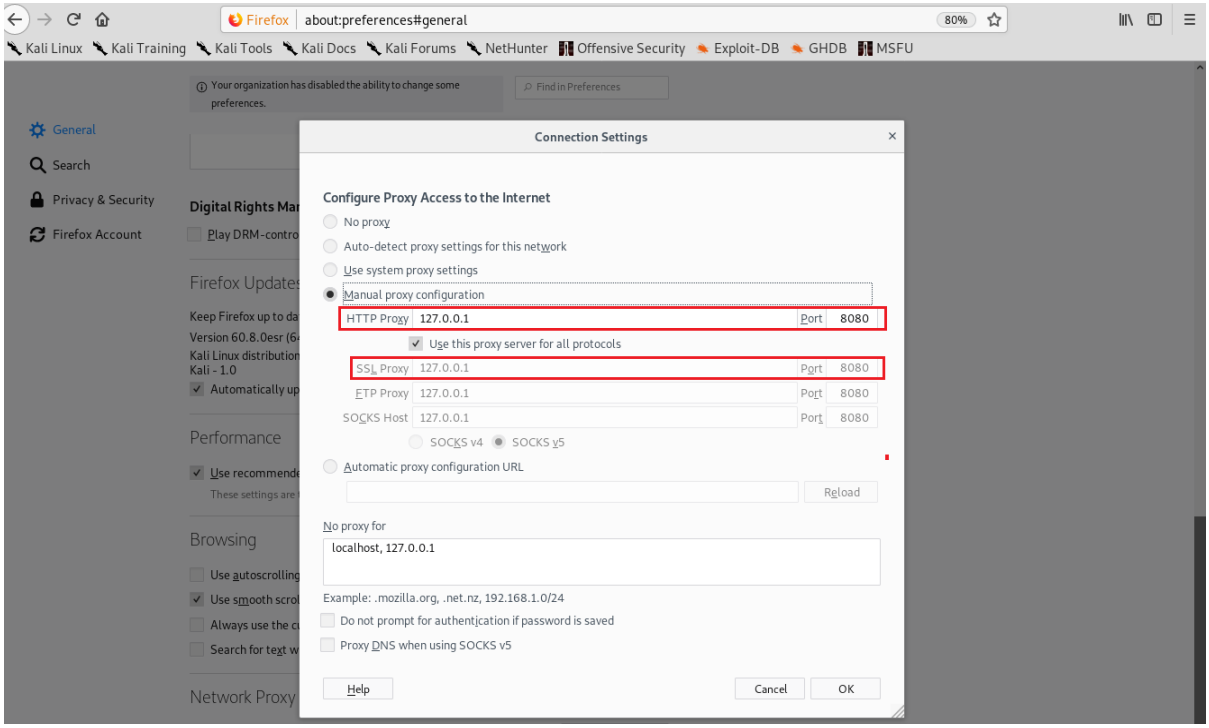


Figura 1 Configuración Proxy

Tras configurar el proxy, se navegará al host http://test/app/bWAPP/login.php y se comprobará que OWASP ZAP recoge todas las peticiones.

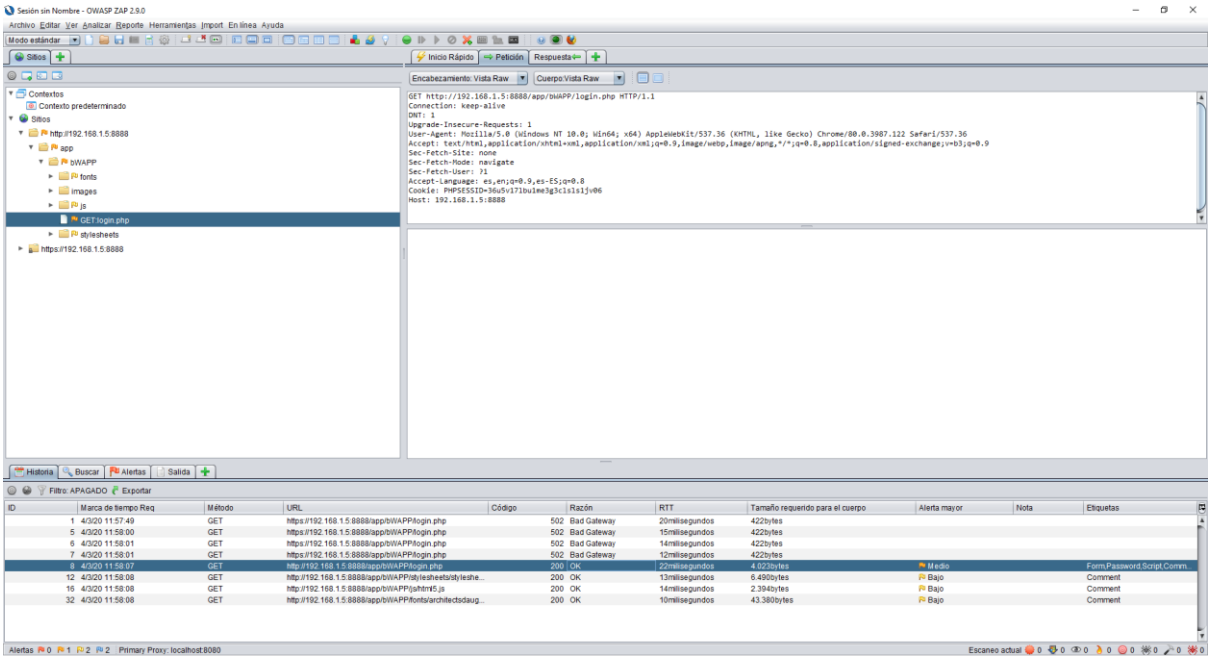


Figura 2 Navegación web del host local "test"

A partir de este momento podemos estudiar el tipo de tráfico intercambiado entre el navegador y el host local, así como las tecnologías empleadas por el mismo.

Por ejemplo, podemos observar de forma inmediata algunas de las cabeceras de seguridad HTTP empleadas por el servidor web.

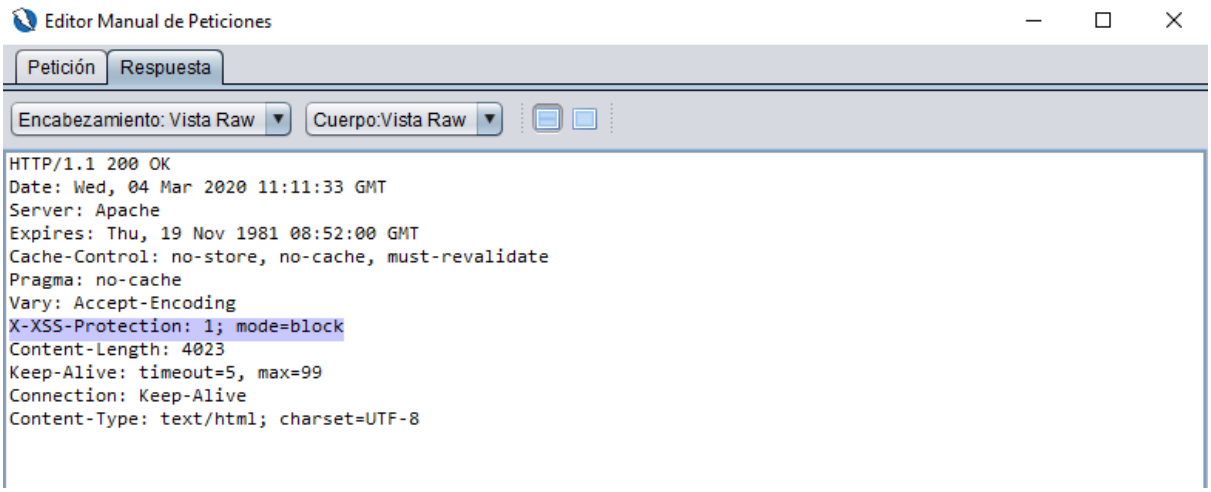
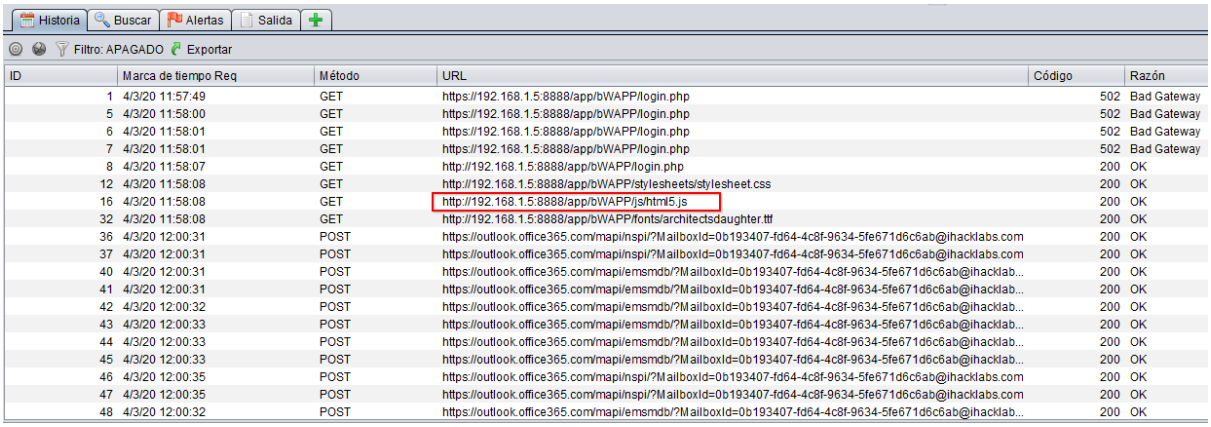


Figura 3 Cabeceras del servicio web

Algunos de estos *headers* (https://owasp.org/www-project-secure-headers/) como, por ejemplo, el uso de *X-XSS-Protection* dificultarían ataques de tipo XSS mediante la activación de determinados filtros en el navegador. Del mismo modo, el *header X-Content-Type-Options* permitiría prevenir cierto tipo de ataques como consecuencia del "*content sniffing*" llevado a cabo por el navegador (http://webblaze.cs.berkeley.edu/papers/barth-

caballero-song.pdf). Se recomienda que el alumno investigue y entienda el uso de estas cabeceras para una mayor comprensión de conceptos relacionados con la seguridad web. En este aspecto, el proyecto OWASP (*Open Web Application Security Project*) es uno de los mejores puntos de partida para comprender los pilares básicos de la seguridad web.

Continuando con el análisis del host "test", en la ventana de información (pestaña Historia) se observa, el tipo de recursos solicitados cuando se accede a la URL principal: CSS, JavaScript, etc.



ID	Marca de tiempo Req	Método	URL	Código	Razón
1	4/3/20 11:57:49	GET	https://192.168.1.5:8888/app/bWAPP/login.php	502	Bad Gateway
5	4/3/20 11:58:00	GET	https://192.168.1.5:8888/app/bWAPP/login.php	502	Bad Gateway
6	4/3/20 11:58:01	GET	https://192.168.1.5:8888/app/bWAPP/login.php	502	Bad Gateway
7	4/3/20 11:58:01	GET	https://192.168.1.5:8888/app/bWAPP/login.php	502	Bad Gateway
8	4/3/20 11:58:07	GET	http://192.168.1.5:8888/app/bWAPP/login.php	200	OK
12	4/3/20 11:58:08	GET	http://192.168.1.5:8888/app/bWAPP/stylesheets/stylesheet.css	200	OK
16	4/3/20 11:58:08	GET	http://192.168.1.5:8888/app/bWAPP/js/html5.js	200	OK
32	4/3/20 11:58:08	GET	http://192.168.1.5:8888/app/bWAPP/fonts/architectsdaughter.ttf	200	OK
36	4/3/20 12:00:31	POST	https://outlook.office365.com/mapi/nspi/?MailboxId=0b193407-fd64-4c8f-9634-5fe671d6c6ab@ihacklabs.com	200	OK
37	4/3/20 12:00:31	POST	https://outlook.office365.com/mapi/nspi/?MailboxId=0b193407-fd64-4c8f-9634-5fe671d6c6ab@ihacklabs.com	200	OK
40	4/3/20 12:00:31	POST	https://outlook.office365.com/mapi/emsmdb/?MailboxId=0b193407-fd64-4c8f-9634-5fe671d6c6ab@ihacklab...	200	OK
41	4/3/20 12:00:31	POST	https://outlook.office365.com/mapi/emsmdb/?MailboxId=0b193407-fd64-4c8f-9634-5fe671d6c6ab@ihacklab...	200	OK
42	4/3/20 12:00:32	POST	https://outlook.office365.com/mapi/emsmdb/?MailboxId=0b193407-fd64-4c8f-9634-5fe671d6c6ab@ihacklab...	200	OK
43	4/3/20 12:00:33	POST	https://outlook.office365.com/mapi/emsmdb/?MailboxId=0b193407-fd64-4c8f-9634-5fe671d6c6ab@ihacklab...	200	OK
44	4/3/20 12:00:33	POST	https://outlook.office365.com/mapi/emsmdb/?MailboxId=0b193407-fd64-4c8f-9634-5fe671d6c6ab@ihacklab...	200	OK
45	4/3/20 12:00:33	POST	https://outlook.office365.com/mapi/emsmdb/?MailboxId=0b193407-fd64-4c8f-9634-5fe671d6c6ab@ihacklab...	200	OK
46	4/3/20 12:00:35	POST	https://outlook.office365.com/mapi/nspi/?MailboxId=0b193407-fd64-4c8f-9634-5fe671d6c6ab@ihacklabs.com	200	OK
47	4/3/20 12:00:35	POST	https://outlook.office365.com/mapi/nspi/?MailboxId=0b193407-fd64-4c8f-9634-5fe671d6c6ab@ihacklabs.com	200	OK
48	4/3/20 12:00:32	POST	https://outlook.office365.com/mapi/emsmdb/?MailboxId=0b193407-fd64-4c8f-9634-5fe671d6c6ab@ihacklab...	200	OK

Figura 4 Histórico de peticiones

Aunque la mayoría de recursos se solicitan vía web podemos filtrar también por métodos POST para localizar parámetros de entada de interés. Por ejemplo, en la siguiente imagen se muestra la información remitida cuando se hace uso de uno de los buscadores de la web. Si tuviéramos la autorización correspondiente para hacer una auditoría de seguridad, sería interesante probar diversos *payloads* con estos parámetros (por medio de la funcionalidad de *fuzzing*) para corroborar si los mismos son susceptibles de alguna vulnerabilidad.

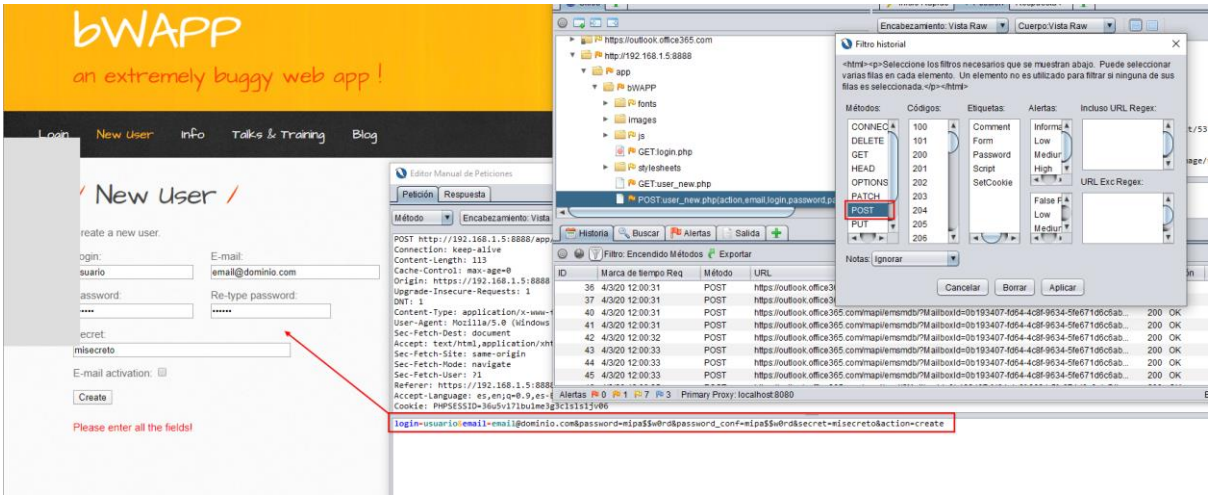


Figura 5 Formulario: peticiones POST

A medida que navegamos consultaremos frecuentemente la ventana de alertas para comprobar si ZAP ha detectado algún problema de seguridad. En la siguiente imagen, por ejemplo, nos informa que se ha detectado un formulario que carece de *token* CSRF (https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html). En este caso, no obstante, la alerta no representa ningún peligro ya que el formulario no está relacionado con la ejecución de acciones potencialmente dañinas o el envío de información crítica.

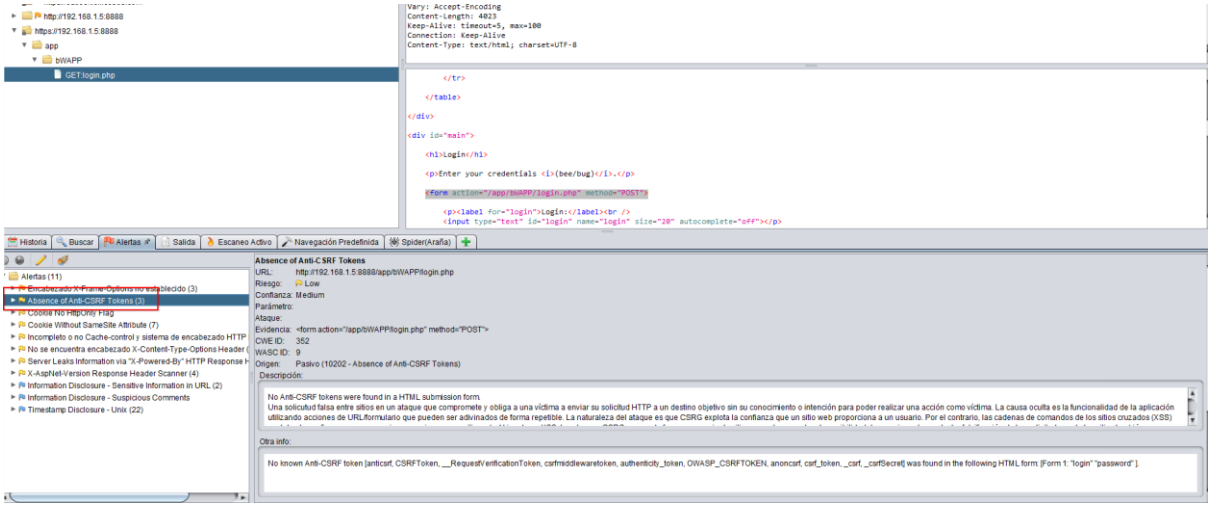


Figura 6 Formulario: peticiones POST

Estudiando de forma meticulosa las peticiones interceptadas por ZAP podremos reconstruir las tecnologías empleadas por el servicio. Asimismo, podremos identificar las entradas susceptibles de ser vulnerables. Recuérdese, no obstante, que para hacer un escaneo activo o utilizar cualquiera de las funcionalidades de ataque disponibles en ZAP se requiere de la autorización correspondiente; en caso contrario podría incurrirse en un delito.

2. EJERCICIO DE INVESTIGACIÓN

El objetivo del ejercicio es investigar el tráfico HTTP empleado por determinado portal web e intentar aprovecharse, si es posible, de una vulnerabilidad en uno de sus parámetros. Debido a que este tipo de pruebas requieren de una autorización previa por parte del responsable del dominio, en su lugar, se instalará un servicio web en una Kali y se realizarán las pruebas en local.

Para proceder con la instalación, descargue el fichero *bWAPP.zip* adjunto y descomprima su contenido dentro del directorio /var/www/. Posteriormente siga las instrucciones descritas en el fichero INSTALL.txt

```
root@kali:/var/www/html/bWAPP# ls -l
total 19652
drwxrwxrwx  2 root root       4096 ene 24 15:12 apps
drwxrwxrwx 13 root root      12288 ene 24 15:12 bWAPP
-rwxrwxrwx  1 root root    5010042 nov  2  2014 bWAPP_intro.pdf
-rwxrwxrwx  1 root vboxsf 15058349 ene 24 15:11 bWAPP_latest.zip
-rwxrwxrwx  1 root root        325 mar  8  2014 ClientAccessPolicy.xml
-rwxrwxrwx  1 root root        200 mar 11  2014 crossdomain.xml
drwxrwxrwx  2 root root       4096 ene 24 15:12 evil
-rwxrwxrwx  1 root root       2589 may 12  2014 INSTALL.txt
-rwxrwxrwx  1 root root       2491 nov  2  2014 README.txt
-rwxrwxrwx  1 root root       8271 nov  2  2014 release_notes.txt
root@kali:/var/www/html/bWAPP# less INSTALL.txt
```

Figura 7 Despliegue bWAPP

Tras la instalación verifique que tiene acceso a la plataforma web desde el navegador y que el mismo está configurado correctamente para utilizar ZAP como proxy web.

Figura 8 Despliegue bWAPP

Posteriormente autentíquese al portal utilizando las credenciales por defecto (si éstas no han sido modificadas):

- **Login**: bee
- **Password**: bug

Una vez autenticado, elija el tipo de *bug* "*OS Command Injection*" y pulse el botón *Hack.*

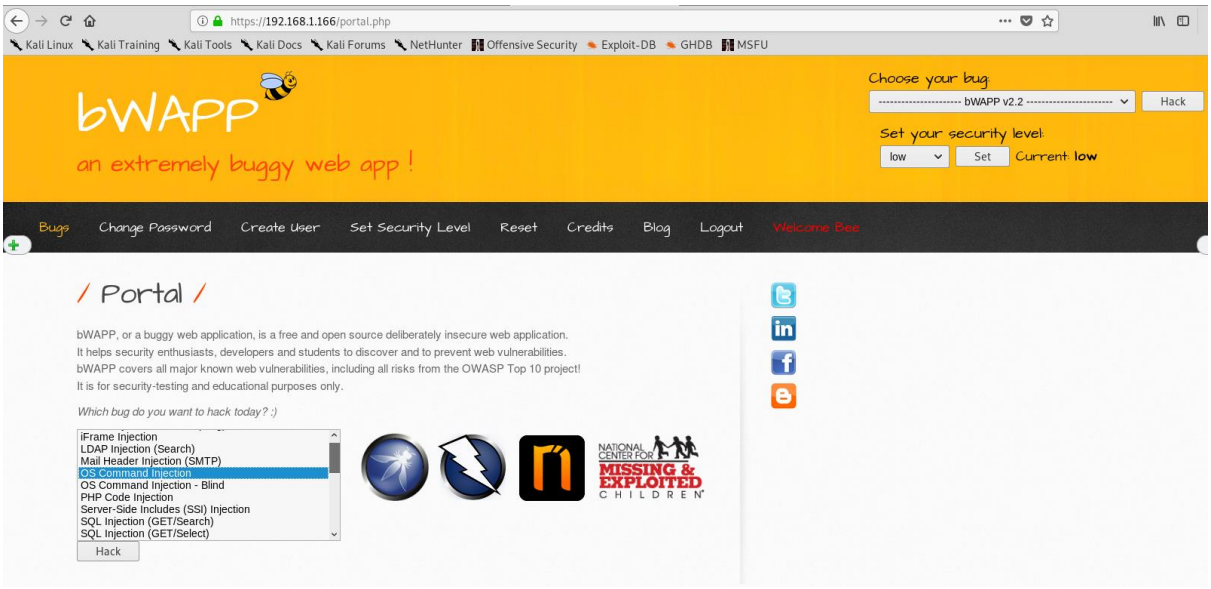


Figura 9 Tipo de prueba: OS Command Injection

El usuario se encontrará con el siguiente portal web donde tendrá que auditar, utilizando ZAP, si alguno de los parámetros utilizados es vulnerable. Asimismo, tendrá que describir el tipo de vulnerabilidad encontrado, sus implicaciones y cómo se solucionaría la misma. Los alumnos no podrán utilizar el escaneo activo ni ninguna de las funciones de ataque implementadas en ZAP (esto incluye: Spider, *fuzzing*, navegación predefinida, etc.). Únicamente se podrá utilizar un enfoque manual con el objetivo de mejorar sus destrezas en el uso de OWASP ZAP. Tras identificar correctamente la vulnerabilidad, los alumnos podrán estudiar el código fuente del script vulnerable.

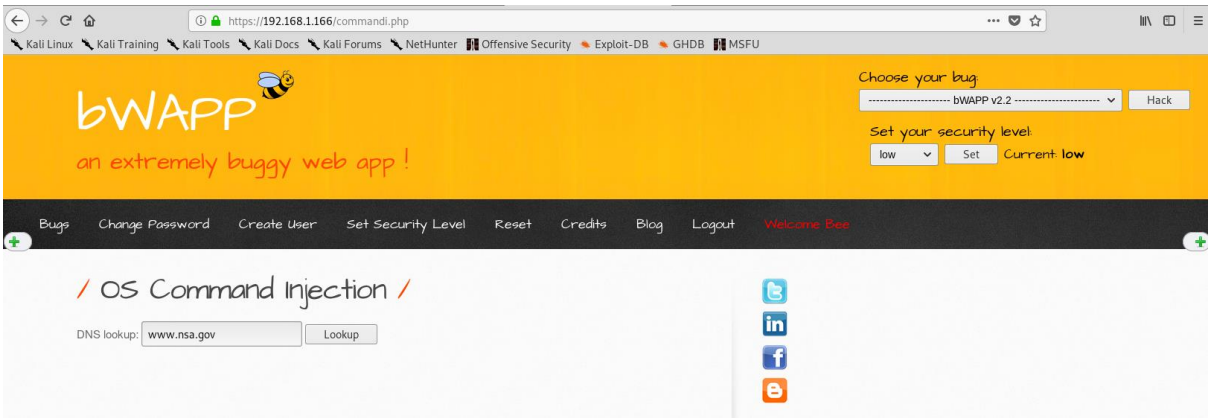


Figura 10 Tipo de prueba: OS Command Injection

Solución del ejercicio:

Si se analiza la petición remitida cuando se pulsa el botón "Lookup", se puede ver que el campo de entrada (por defecto, el valor www.nsa.gov) se remite vía POST mediante el parámetro *target.*

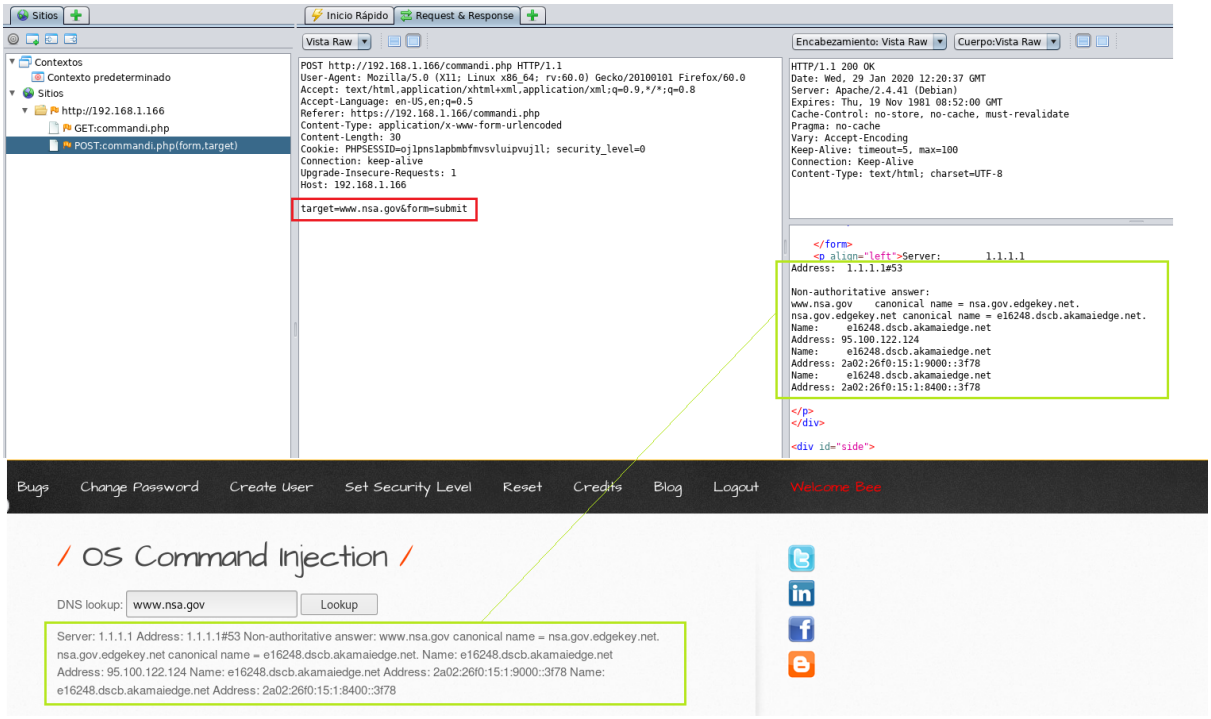


Figura 11 Envío petición POST

Fíjese que el resultado devuelto por el servidor encaja perfectamente con la salida generada por el comando ***nslookup*:**

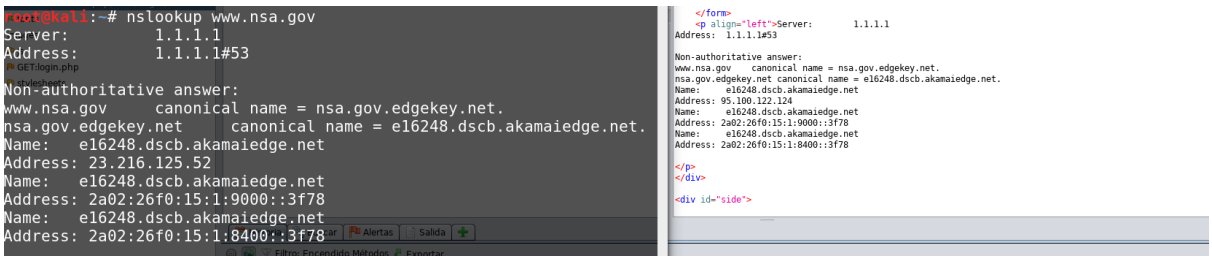


Figura 12 Similitudes con el comando nslookup

Todo parece indicar que el servidor web está utilizando el valor remitido por el parámetro *target* para ejecutarlo en la consola y devolver la salida generada. ¿Qué pasaría si en lugar de remitir el dominio añadimos un segundo comando en la entrada? Por ejemplo, la orden: `;ls`

Para modificar al vuelo dicho argumento vamos a hacer uso de los puntos de interrupción descritos en el curso con el que podremos "parar" la petición web antes de que ésta se remita al servicio web.

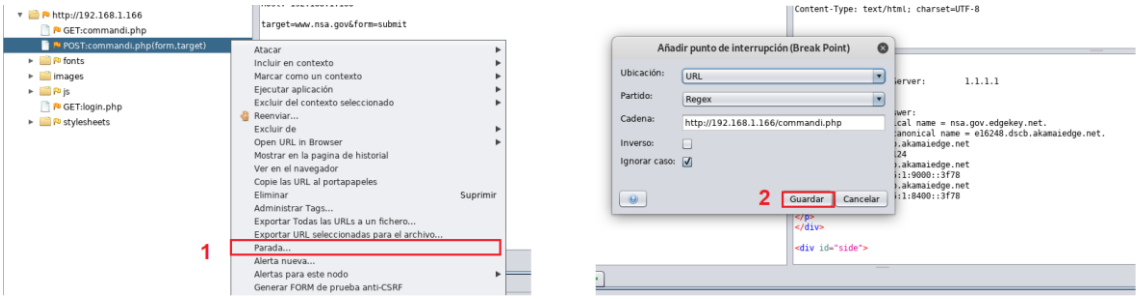


Figura 13 Punto de interrupción

Tras crear el punto de interrupción volveremos a hacer clic en el botón “Lookup” y sustituiremos el contenido del parámetro “target” con la siguiente cadena:

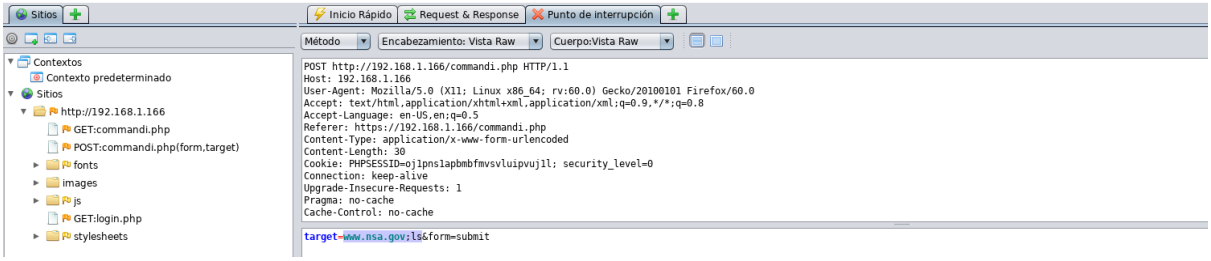


Figura 14 Modificación del parámetro target

Fíjese que de esta manera se ejecutaría la orden: nslookup www.nsa.gov;ls (en Linux el carácter ‘;’ permite concatenar varios comandos). Tras reenviar la petición al destino podemos observar que efectivamente hemos logrado recuperar el listado de ficheros del servidor lo que confirmaría el tipo de vulnerabilidad.

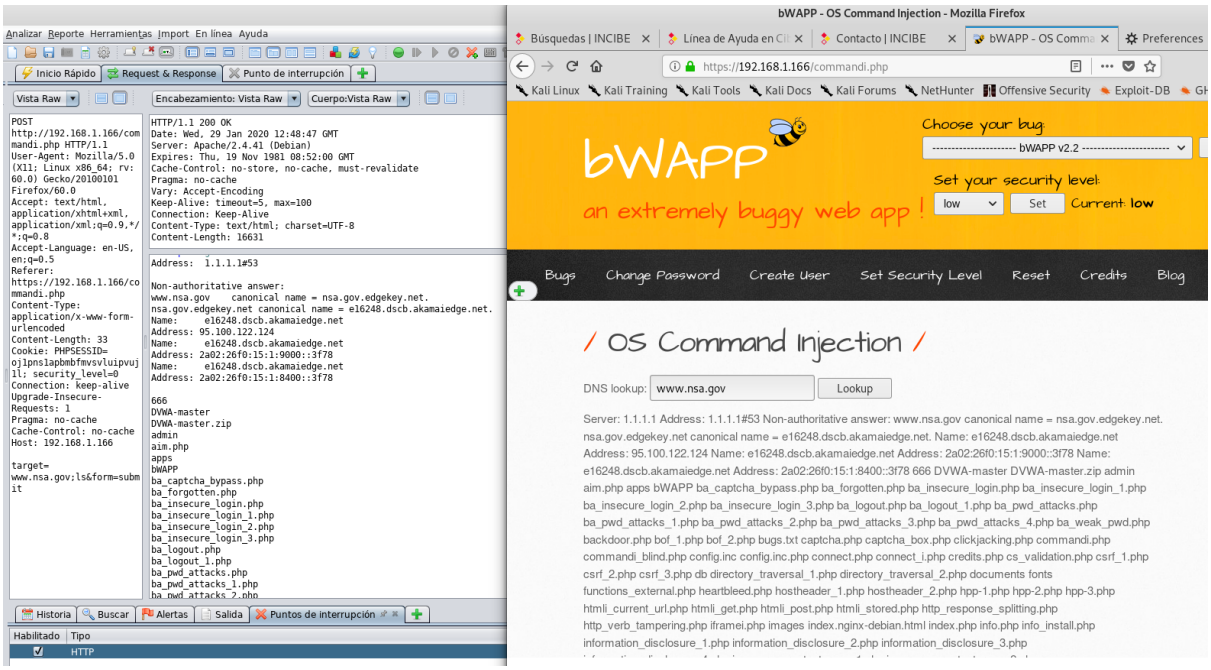


Figura 15 Confirmación de la inyección

Con esta información podemos contestar a las siguientes preguntas:

2.1. ¿Qué tipo de vulnerabilidad se ha identificado?

La vulnerabilidad se corresponde con una inyección de comandos: según describe la OWASP (https://owasp.org/www-community/attacks/Command_Injection), este tipo de ataque:

"es posible cuando una aplicación pasa datos inseguros proporcionados por el usuario (formularios, cookies, encabezados HTTP, etc.) a un shell del sistema. En este ataque, los comandos del sistema operativo proporcionados por el atacante generalmente se ejecutan con los privilegios de la aplicación vulnerable. Los ataques de inyección de comandos son posibles en gran medida debido a una validación de entrada insuficiente"

Si accedemos al directorio web y abrimos el script *commandi.php* podemos corroborar que el *script* está invocando el comando *nslookup* por medio de la función insegura *shell_exec* (https://www.php.net/manual/es/function.shell-exec.php) sin aplicar ningún tipo de validación ni filtro al parámetro *target.*

```
        <label for="target">DNS lookup:</label>
        <input type="text" id="target" name="target" value="www.nsa.gov">

        <button type="submit" name="form" value="submit">Lookup</button>

        </p>

    </form>
    <?php

    if(isset($_POST["target"]))
    {

        $target = $_POST["target"];

        if($target == "")
        {

            echo "<font color=\"red\">Enter a domain name...</font>";

        }

        else
        {

            echo "<p align=\"left\">" . shell_exec("nslookup  " . commandi($target)) . "</p>";

        }

    }

    ?>

</div>

<div id="side">

    <a href="http://twitter.com/MME_IT" target="blank_" class="button"><img src="./images/twitter.png"></a>
    <a href="http://be.linkedin.com/in/malikmesellem" target="blank_" class="button"><img src="./images/linkedin.png"></a>
    <a href="http://www.facebook.com/pages/MME-IT-Audits-Security/104153019664877" target="blank_" class="button"><img src="./images/facebook.png"></a>
```

Figura 16 Código fuente (commandi.php)

2.2. ¿Qué tipo de implicaciones tiene?

Un atacante podría ejecutar todo tipo de órdenes y, por tanto, comprometer al completo el servidor web. Por ejemplo, si en lugar de ejecutar un "ls" el atacante hubiera ejecutado "*nc -l -p 2222 -c /bin/bash"* instalaría una *bind shell* a modo de backdoor.

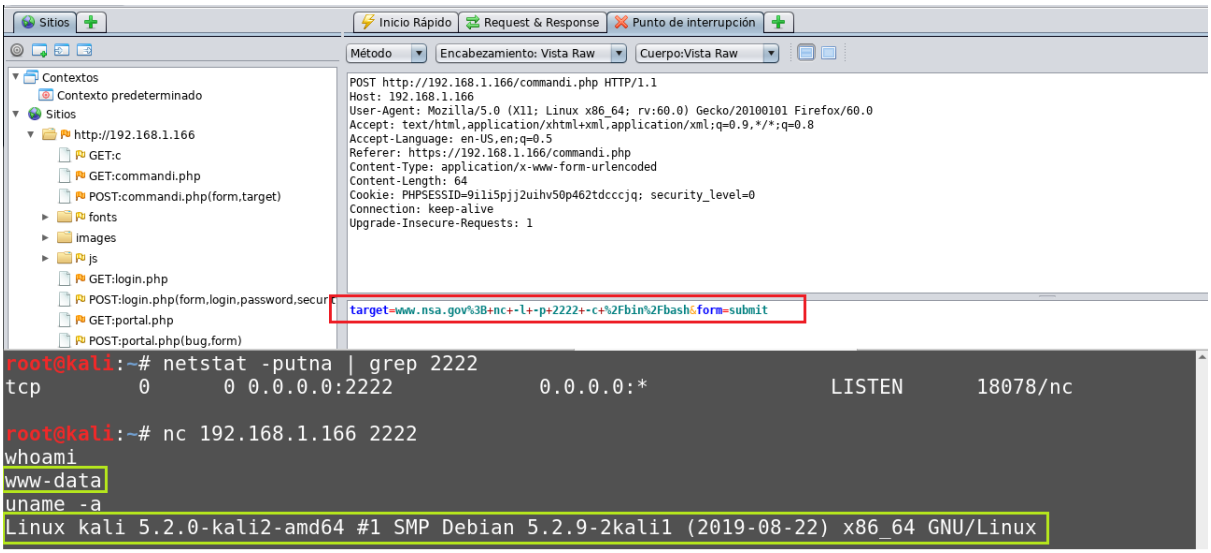


Figura 17 Bind shell con netcat

2.3. ¿Cómo solucionaría la vulnerabilidad?

Se recomienda utilizar las pautas descritas por el proyecto OWAS (https://cheatsheetseries.owasp.org/cheatsheets/OS_Command_Injection_Defense_Cheat_Sheet.html). Estas contramedidas se resumen en:

- Evitar invocar directamente comandos.
- Escapar y filtrar los valores proporcionados a los comandos.
- Parametrización junto con la validación apropiada de los parámetros de entrada.

3. EJERCICIO ADICIONAL

Aprovechando la instalación del servicio web local utilizado en el apartado de investigación anterior, en este ejercicio los usuarios aprenderán a incorporar nuevos diccionarios a ZAP para el descubrimiento de nuevos recursos web.

Un repositorio de especial interés para este ejercicio es el proyecto en Github Fuzzdb ya que recoge multitud de diccionarios dentro del directorio *predictable-filepaths* (https://github.com/fuzzdb-project/fuzzdb/tree/master/discovery/predictable-filepaths) para multitud de tecnologías web; por ejemplo, para CMS (Drupal, Joomla, WordPress, etc.), archivos de login comúnmente utilizados para diversas plataformas, etc.

Si queremos hacer uso de estos diccionarios podemos descargarlos manualmente desde Github o bien clonar todo el repositorio en local.

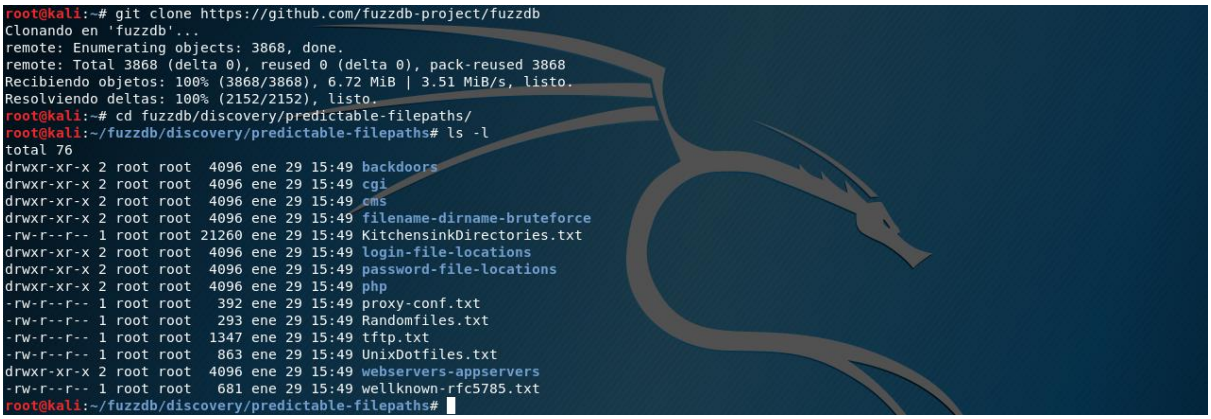


Figura 18 Descarga de diccionarios

Posteriormente, si queremos añadir algunos de estos diccionarios a ZAP, para poder ser utilizados con la funcionalidad “Navegación predefinida”, nos dirigiremos al menú “Herramientas -> Opciones” y posteriormente seleccionaremos el diccionario que queremos (en la siguiente imagen se ha seleccionado el diccionario *raft-large-directories.txt*). Fíjese que desde este menú podemos configurar también si deseamos incluir la navegación de ficheros, las extensiones que deseamos incluir y otros parámetros relacionados con el rendimiento, por ejemplo, el número de hilos a utilizar. Si queremos integrar más diccionarios repetiremos el mismo proceso.

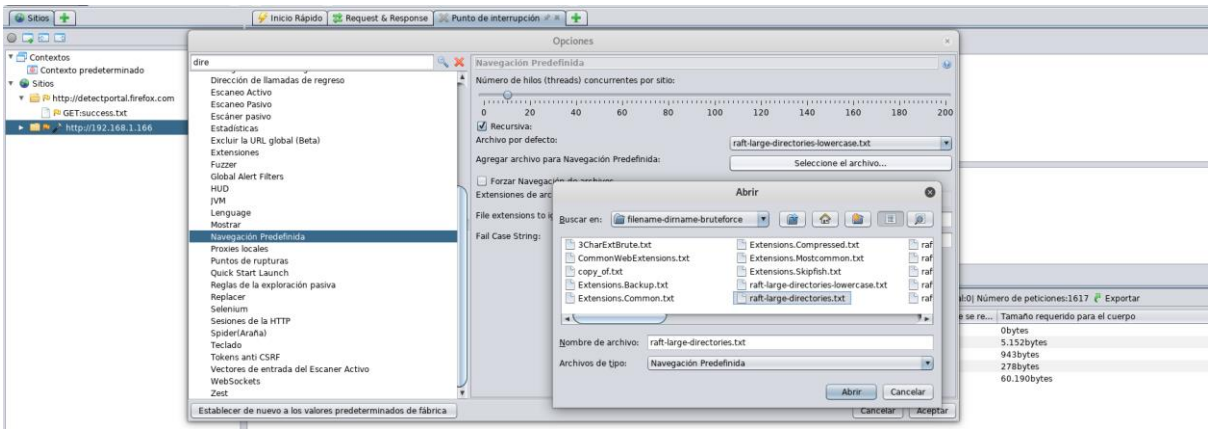


Figura 19 Incorporación de nuevos diccionarios

Una vez añadidos los diccionarios, seleccionaremos el recurso a partir del cual deseamos descubrir nuevos directorios y, haciendo clic con el botón derecho, seleccionaremos la opción de “Directorio de navegación definido” dentro del menú Atacar. Fíjese que en la ventana inferior aparecerán los diccionarios previamente añadidos y podremos seleccionar cualquiera de ellos.

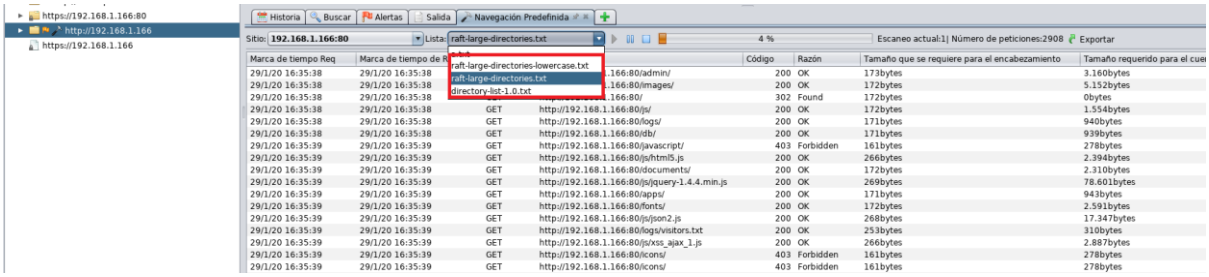


Figura 20 Incorporación de nuevos diccionarios

Tal y como se detalló en el seminario web, el descubrimiento de directorios y ficheros por medio de la funcionalidad “Navegación predefinida” es de gran utilidad para identificar recursos no referenciados. En ocasiones, estos recursos nos permiten acceder a directorios que, por error o descuido, se han hecho públicos y que ofrecen información sobre la plataforma, tecnologías utilizadas o cualquier otro tipo de datos sensible sobre la configuración del servicio web. En la siguiente imagen se muestra uno de los ficheros de configuración identificados gracias a uno de los diccionarios; el fichero “config.inc” situado en el directorio raíz. Fíjese que el mismo incluye credenciales de acceso a cierta base de datos.

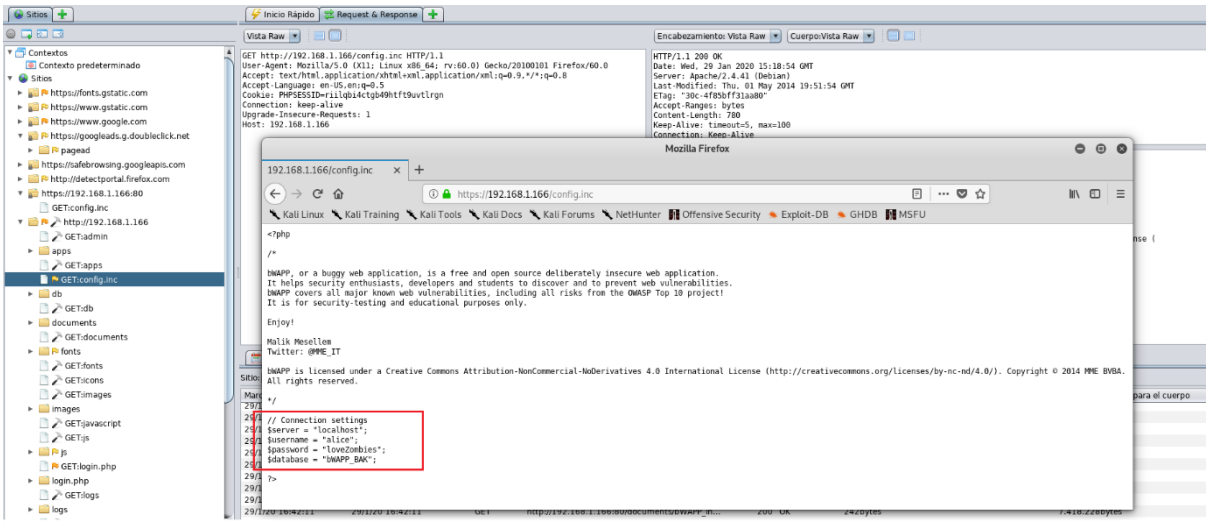


Figura 21 Identificación del fichero de configuración "config.inc"