Owasp Zed Attack Proxy Guide

Eric Balderrama

Published with GitBook

Tabla de contenido

Owasp Zed Attack Proxy Guide

Autor: Balderrama Eric

Twitter: @EricBalderrama

Editado por: Jose Moruno Cadima @sniferl4bs

Sitio Web: www.sniferl4bs.com

Licencia

OWASP Zed Attack Proxy Guide.

Esta obra está sujeta a la licencia Reconocimiento-CompartirIgual 4.0 Internacional de Creative Commons. Para ver una copia de esta licencia, visite http://creativecommons.org/licenses/by-sa/4.0/.

Puede hallar permisos más allá de los concedidos con esta licencia en http://www.sniferl4bs.com

Todas las marcas son propiedad de sus respectivos dueños.

Imagen de la portada realizada por @GONZALO CABRERA

Sugerencias y comentarios a sniferl4bs@gmail.com

Garantía

Este Ebook se distribuye con la esperanza de que sea útil y sirva como una referencia para el uso de la herramienta ZAP de OWASP. El autor no asume ninguna responsabilidad si el lector hace un mal uso del mismo.

Agradecimientos

Dar las gracias a todos los que visitan el blog, como tambien a los que comparten contenido y generan

Sobre el autor

Introducción a Zaproxy

La herramienta que conoceremos en este E book es utilizada para el análisis web la cual tiene una asombrosa versatilidad y características que pueden resultar de gran ayuda al momento de realizar una auditoría.

Hablaremos de Zed Attack Proxy (ZAP), un proyecto desarrollado por la comunidad de OWASP y cuyo lider de proyecto es Simon Bennetts.

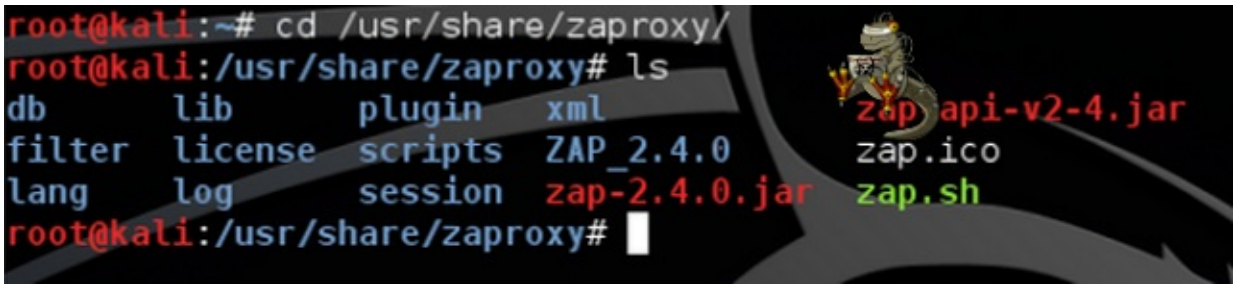
La url del proyecto es: OWASP Zed Attack Proxy Project

Todas las pruebas que haremos con Zap a lo largo del libro serán llevadas a cabo sobre Kali Linux, no es requisito indispensable que virtualicen o instalen un Kali ya que Zap está desarrolado en java y por ende es multiplataforma.

La descarga puede ser realizada desde Github, allí se encuentran las versiones para los distintos Sistemas Operativos (Linux, Windows y Mac) y el core que contiene funciones mínimas. De todos modos, la versión para Linux es cross plataform, recomiendo bajar esa.

El único requisito que tiene es tener la versión 7 de java o superior.

Comencemos a ver de qué se trata (utilizaremos la versión 2.4.0 para Linux desde un Kali), ingresemos a */usr/share/zaproxy* y luego listemos para ver los archivos que tiene dentro:



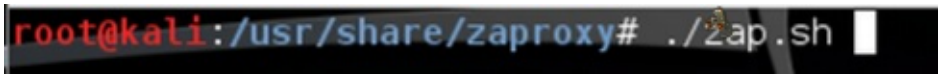
Aquí podemos encontrar las base de datos que utiliza, filtros. lenguajes, librerías, scripts, etc. Es interesante analizar el contenido de cada directorio, pero eso queda en sus manos.

Lo que nos interesa son dos archivos, el zap.sh para correr desde plataformas Linux y zap-2.4.0.jar para poder correr desde cualquier SO.

Para inicializar desde el jar:

Se puede utilizar el mismo comando para inicializarlo desde windows.

Para inicializar desde el script:



Para inicializar desde el menú:

Cualquiera de las formas que iniciemos el Zap está bien, una vez levante el GUI, nos encontraremos con una aplicación muy intuitiva y sencilla de usar:

En el campo “URL to attack” ingresamos el target a analizar, en éste primer caso analizaremos “http://scanme.nmap.org”.

Cabe aclarar que siempre debemos ingresar el protocolo, es decir si el sitio que queremos analizar se comunica por **http** o por **https**, si ingresamos solo e nombre de dominio no mostrará el siguiente error:

En la próxima sección analizaremos los output que nos entrega como resultado del análisis y comenzaremos con algunas pruebas.

Intercept Proxy

En este capítulo de ZAP nos enfocaremos en sus funcionalidades de intercepción por proxy.

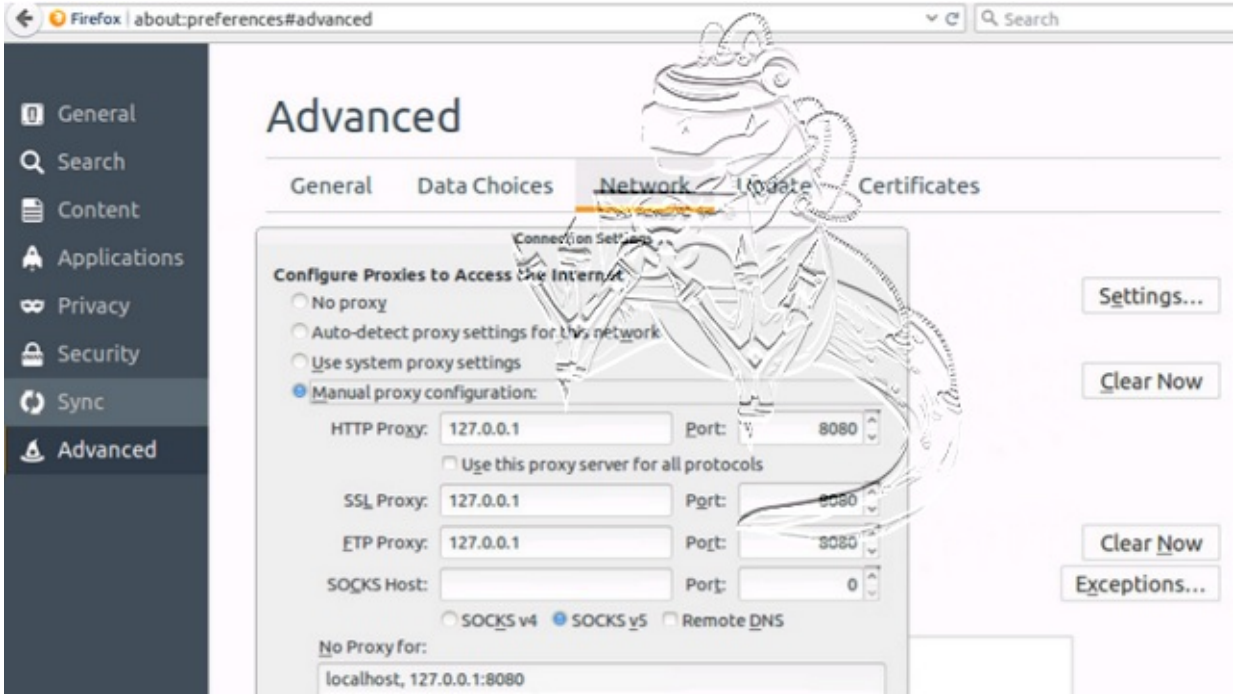
Como ya vimos en el capitulo anterior, ZAP es una tool de pentesting de web applications y como tal tiene la funcionalidad de ser utilizado como web proxy entre otras tantas que ya iremos viendo.

Antes de que levanten ZAP, vamos a tener que hacer unas configuraciones en nuestro navegador, lo haremos en Firefox, pero pueden utilizar el de su eleccion, en todos es bastante similar la configuracion.

Preferences -> advanced -> network -> Settings

Si quieren acortar camino, abran una pestaña y peguen: *about:preferences#advanced* luego solo ir a settings o configuración y verán el panel de **'configuración de conexiones'**.

En éste panel, debemos seleccionar **'Manual proxy configuration'** y en los campos de HTTP, SSL y FTP Proxy debemos colocar nuestra dirección de localhost *127.0.0.1* y puerto *8080*. En el campo '*No Proxy for*' debemos declarar localhost y 127.0.0.1:8080 separados por una coma (,) como se ve en la imagen.

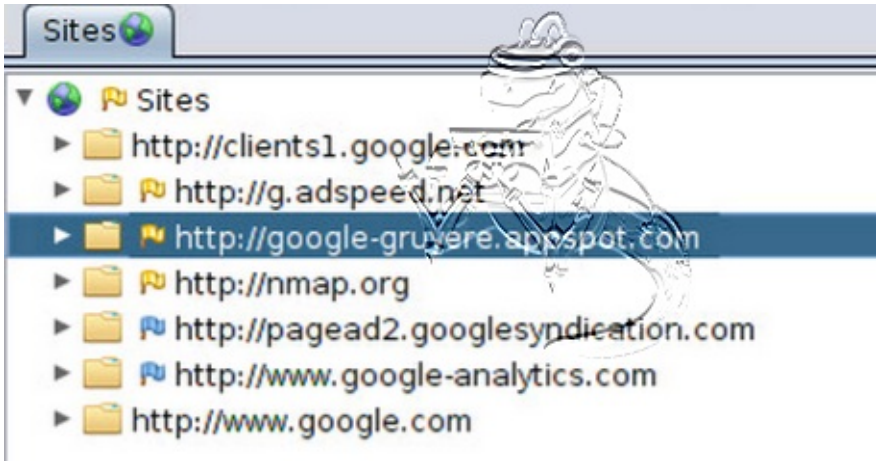


También se puede usar la configuración que se realiza con FoxyProxy como se realizo en BurpSuite es aplicable con Zap.

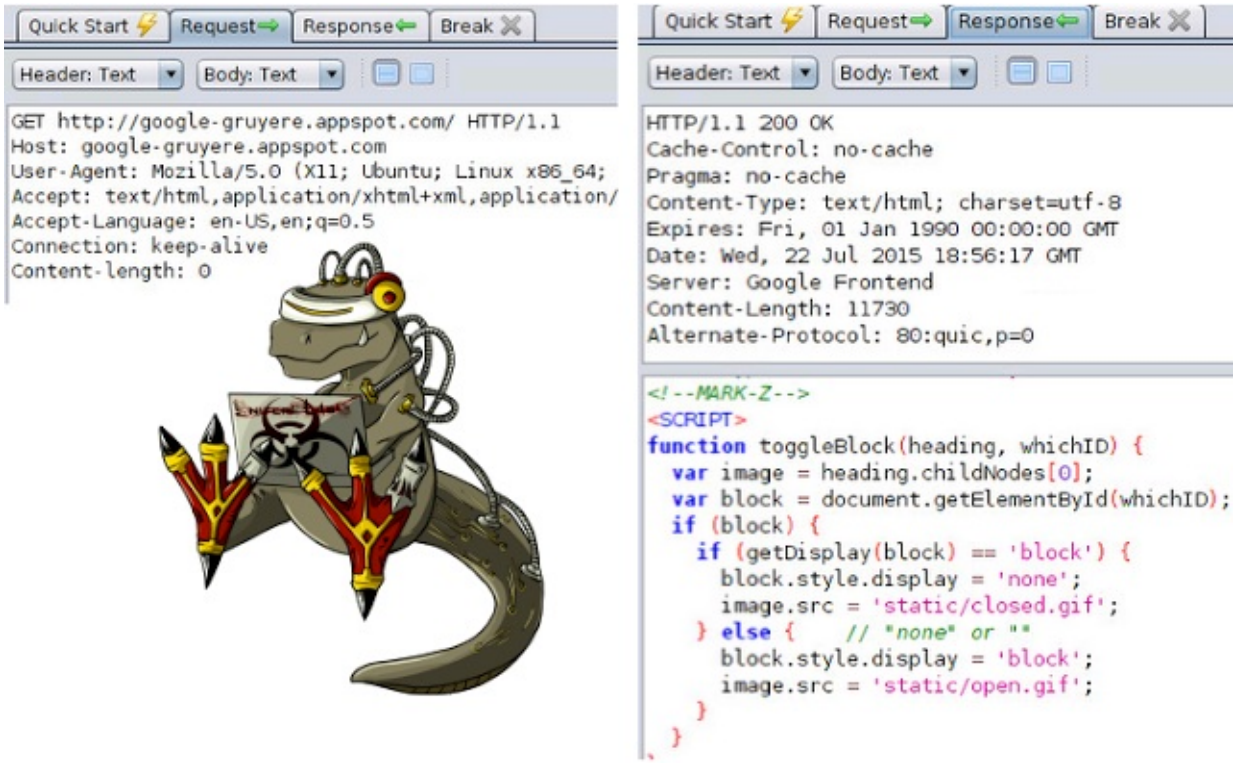
Ahora, sólo basta con ingresar a un website para que Zap comience a trabajar.

Paneles y principales características

Tras inicializar la herramienta, accederemos a "Google Gruyere" desde el navegador que esta configurado, volvemos a zap y observamos en el panel "*Sites*" que ya empezó a guardar los sitios que visitamos.

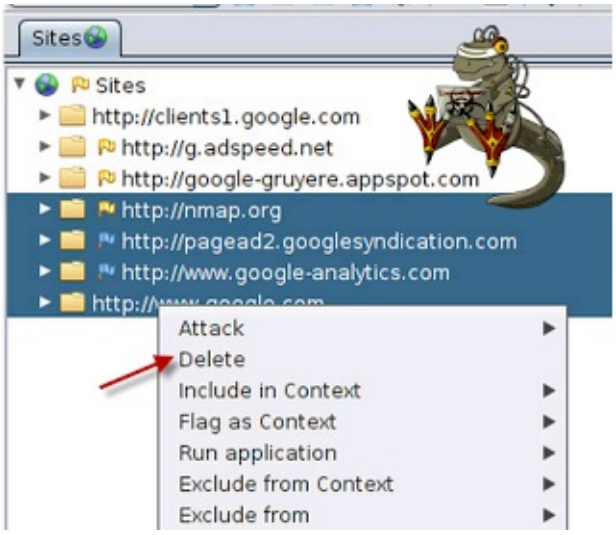


En la parte derecha, se observa los *"Request"* y los *"Response"* del sitio en el momento que se ha accedido.

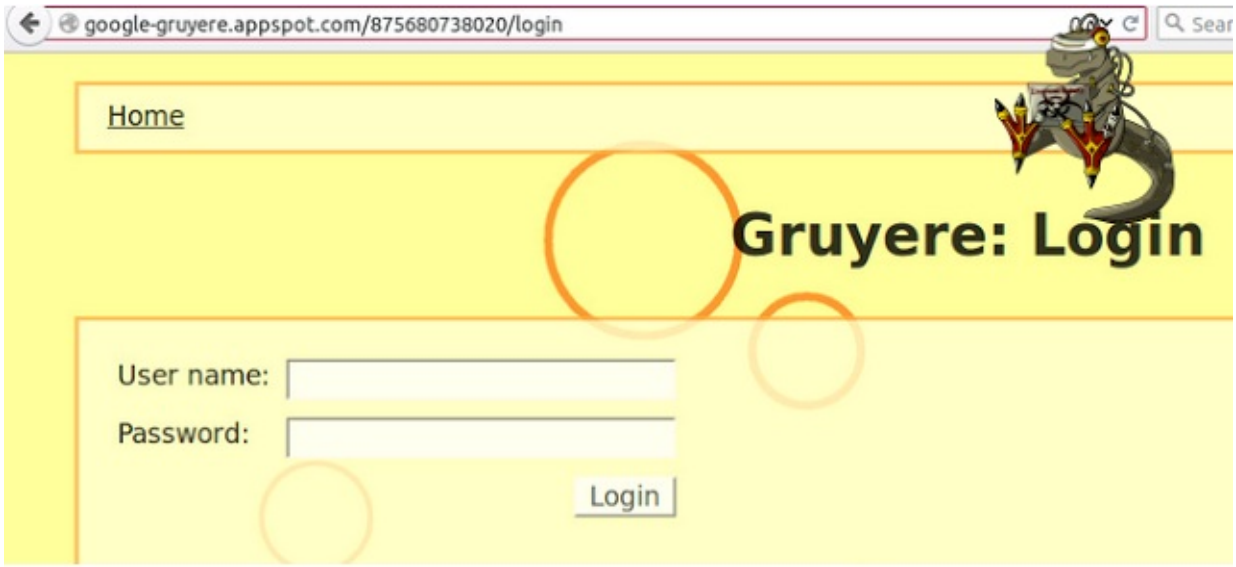


Si queremos ver en detalle los Request y Response entre nuestro navegador y el nodo, podemos ir a las pestañas indicadas, desde la pestaña Response por ejemplo, logramos visualizar los values que solicita el login.

Si dado el caso, hemos recorrido varios sitios y se nos ha llenado el panel de Sites de nodos, podemos ir eliminando los que no nos interesan para tener un ambiente de trabajo más organizado y limpio. Solo basta con seleccionar las url/nodos que no nos interesan, click derecho y “delete”.

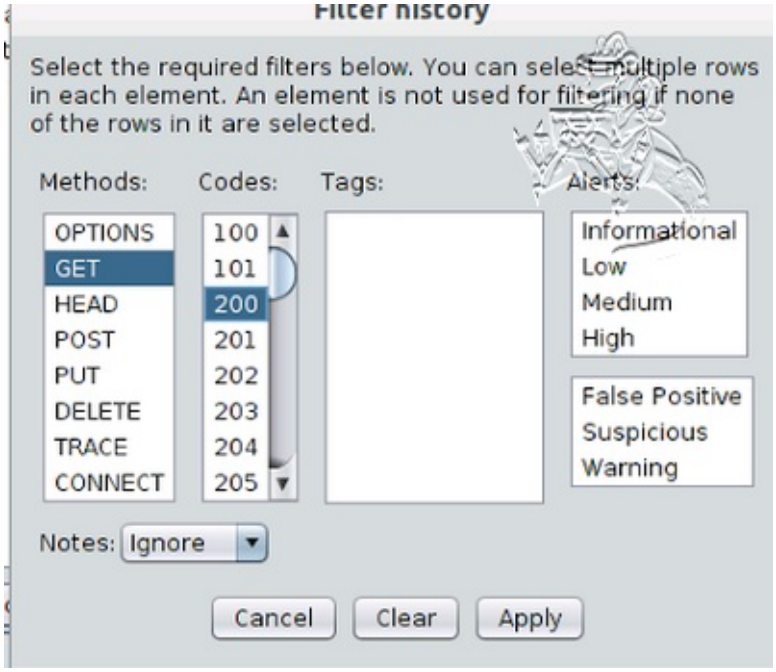


Desde el browser accedemos a “google-gruyere.appspot.com/875680738020/login”, un panel de login común:



Desde ZAP nos movemos por los directorios que fue relevando con sus solicitudes y respuestas en el panel de *"Sites"*, abajo, en *"History"* también se visualiza la petición por GET que se hizo al path de login, tiempo de respuesta y qué devolvió. Allí podemos filtrar por métodos de peticiones, codes, tags y alertas. Aquí vemos un 302.

Si aplicamos los filtros "GET + 200" que se ven en la imagen

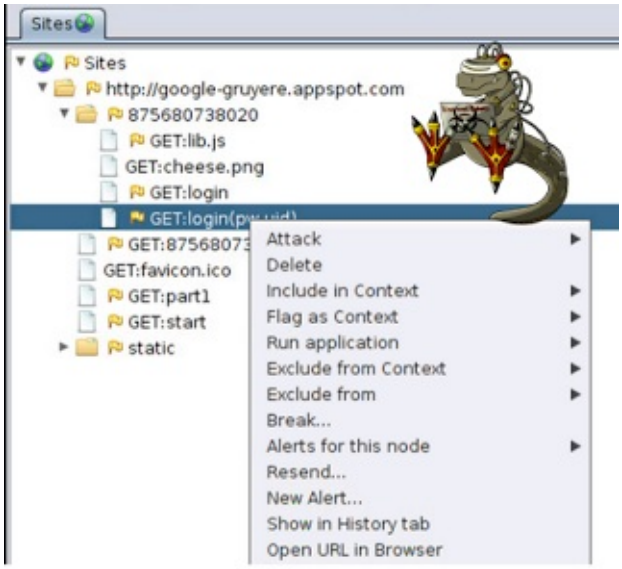


Sólo veremos los GET que nos hayan respondido un 200 OK:

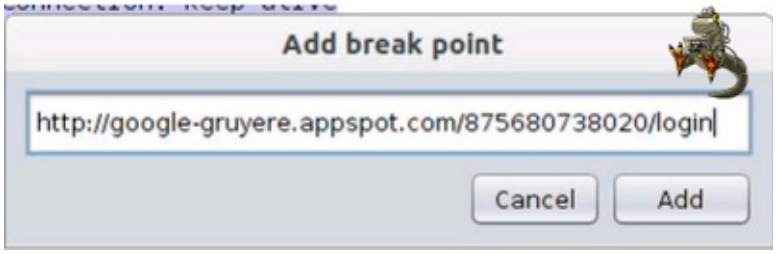
Usando break points

Hasta aquí utilizamos zap para que intercepte todo sitio por el que nos movamos, ahora bien, al utilizar break points tendremos el control de stopear o fowardear las peticiones entre browser y server.

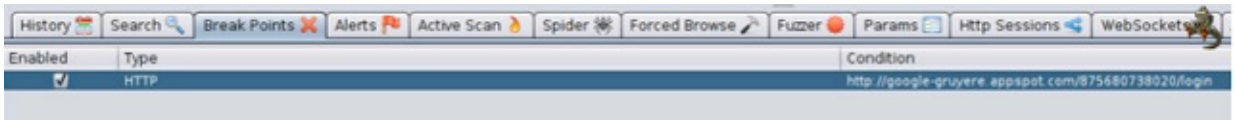
Si hacemos click derecho sobre el target y luego seleccionamos Break



Se abrirá una ventana en la cual definimos la url y agregamos el break.



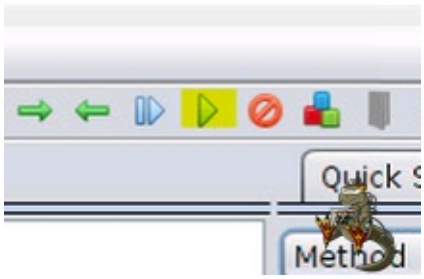
Para comprobar si hemos colocado bien el break point vamos a la pestaña que se encuentra abajo con el mismo nombre:



Al ingresar al sitio desde el browser y colocar las credenciales de autenticación veremos que queda cargando y no resolverá hasta que forwardemos la petición desde el Zap.

Al ir a zap, vemos que ha interceptado el envío de las credenciales

Para que la petición siga su camino debemos debemos hacer click en el icono de play.

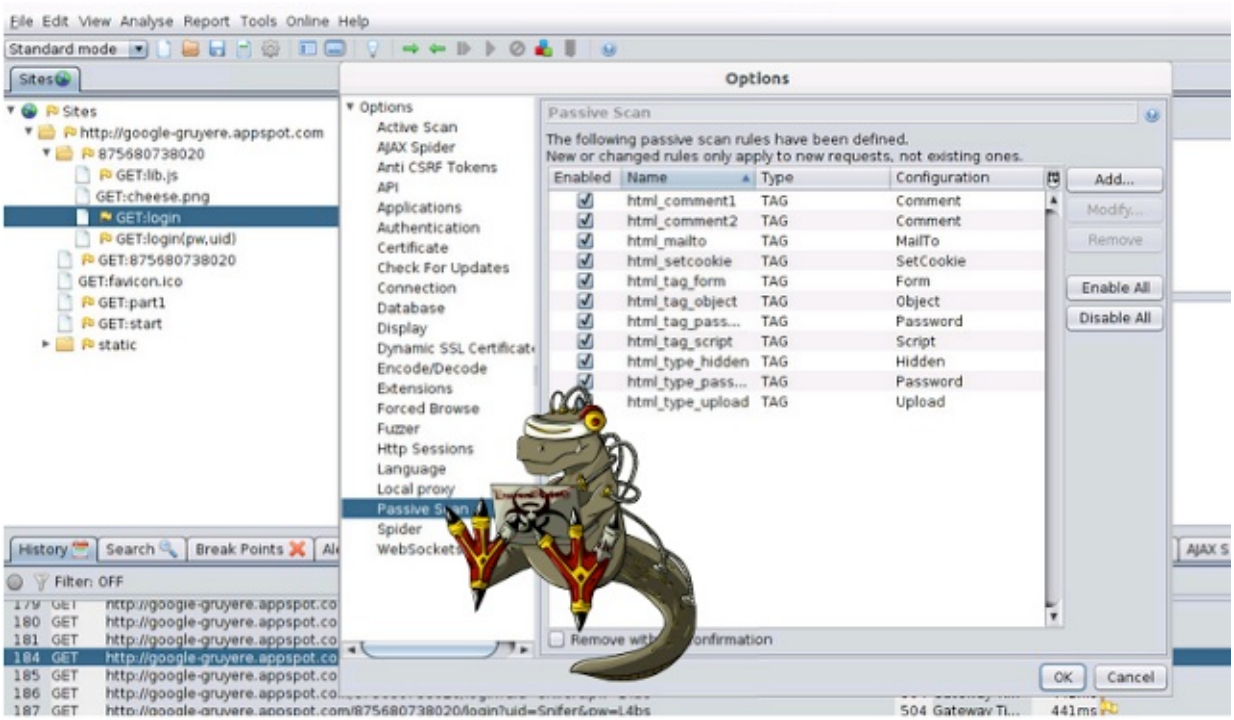


La pantalla que nos muestra la pestaña “Break” se pondrá en blanco y la petición será enviada.

Escaneos Pasivos y Activos

Zap, además de funcionar como proxy, también tiene la capacidad de servirnos de scanner. Posee dos modalidades de scanning, la pasiva y la activa, en esta sección profundizaremos en passive scanning.

El escaneo pasivo sólo intercepta las respuestas del server y no es intrusivo. Sus reglas están disponible en : *tools - options - passive scan*



Desde aquí podemos elegir las reglas a utilizar, editarlas, crear nuevas o eliminar las que no utilizamos.

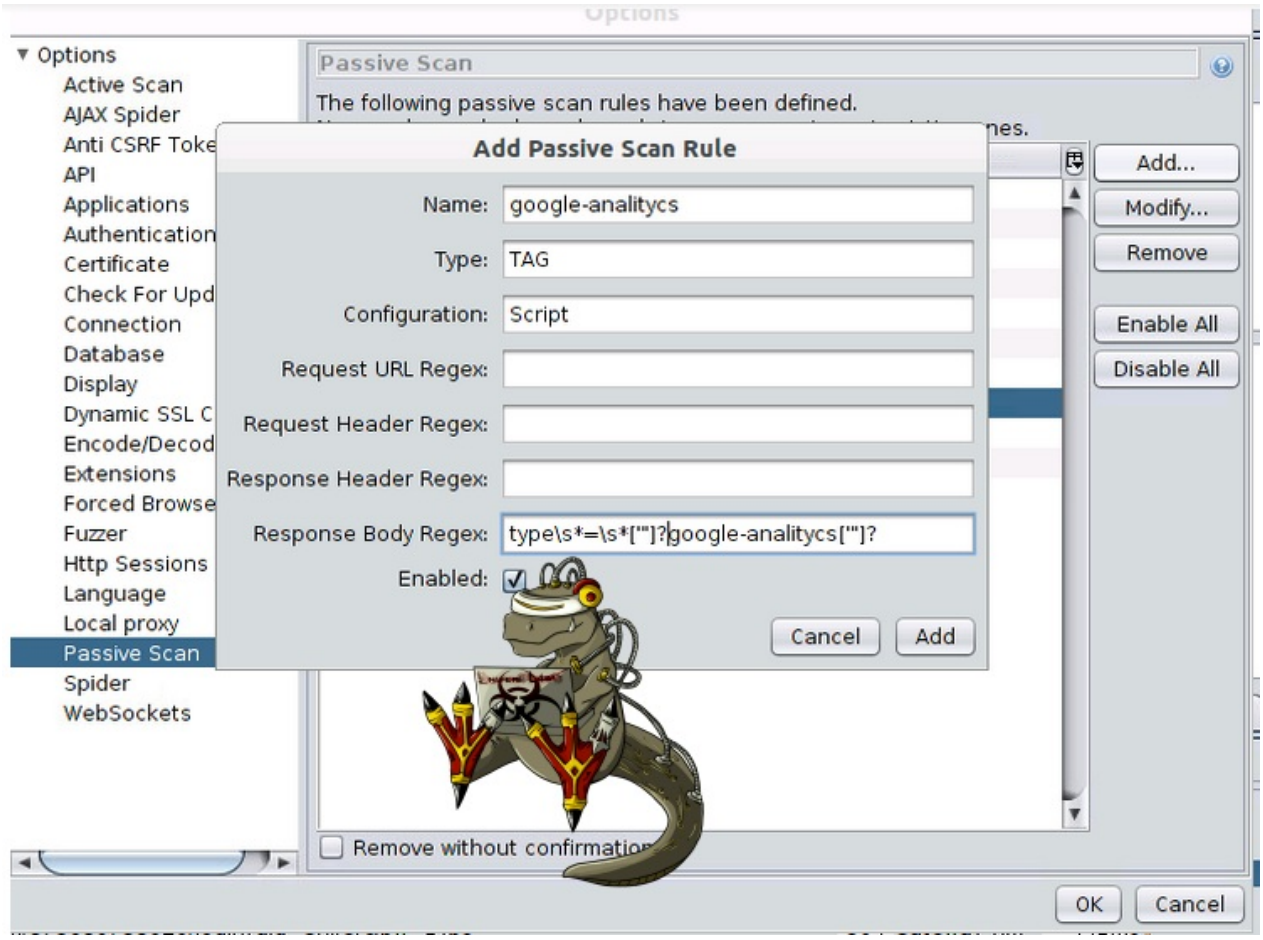
Si deseamos agregar una nueva regla vamos a add. Es recomendable analizar las reglas ya existentes para entender la lógica de su estructura.

Hagamos un ejemplo para identificar la existencia de google analitycs en los sitios que analizaremos:

Name: lo que quieras **Type:** TAG viene por defecto **Configuration:** lo que desees **Response Body Regax:** aqui es donde debemos colocar que encuentre nuestro string “google-analitycs”, ya que en el fuente se ve de la siguiente manera:

```
(function() {
  var ga = document.createElement('script'); ga.type = 'text/javascript'; ga.async = true;
  ga.src = ('https:' == document.location.protocol ? 'https://ssl' : 'http://www') + '.google-analytics.com/ga.js';
  var s = document.getElementsByTagName('script')[0]; s.parentNode.insertBefore(ga, s);
})();
```

Nuestra configuración entonces quedaría algo así:



Anteponer con un **"\"** antes del **"*"** implica la capacidad de poder citar caracteres especiales y el **"*"** en expresiones regulares significa corresponder lo precedente cualquier número veces.

Tras establecer las reglas que utilizaremos en el escaneo, procedemos a poner el zap como proxy intercepter como lo vimos en el capítulo anterior.

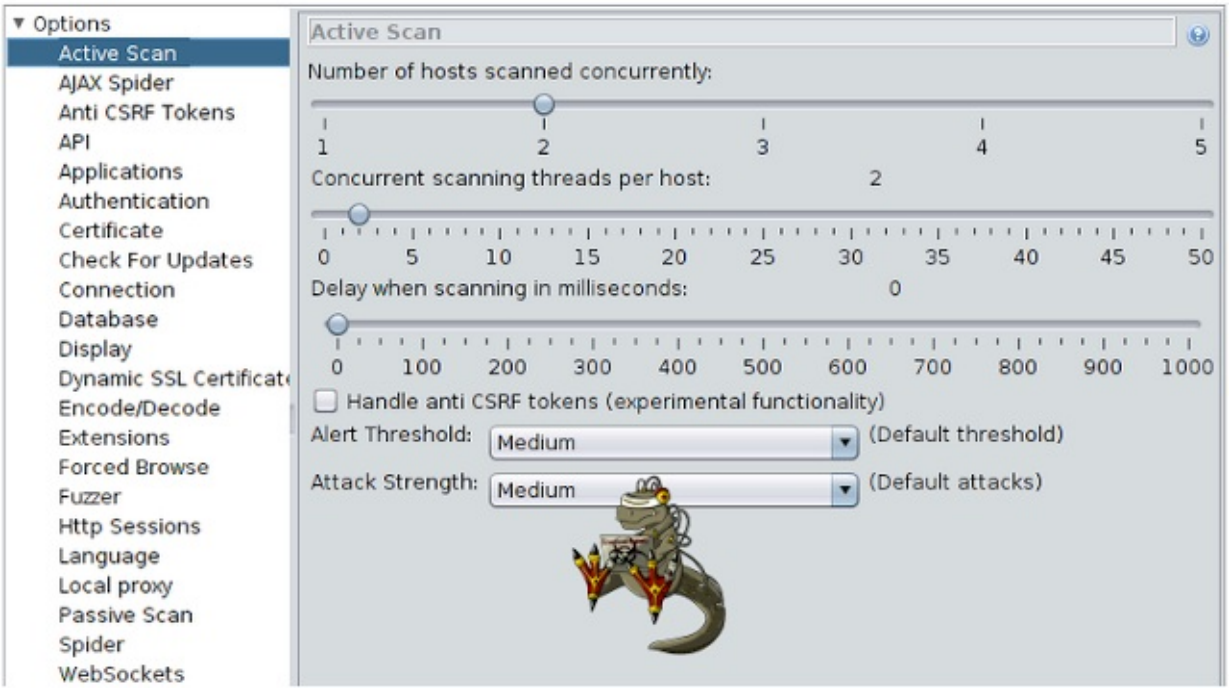
Una vez que tengamos suficientes request podemos usar los filtro (que también vimos) para focalizar nuestro análisis.

Las reglas por defecto incluidas en el escaner pasivo de ZAP incluyen la capacidad de detectar comentarios, direcciones de correos electrónicos, cookies, formularios, objetos, contraseñas, scripts, campos ocultos, entre otras.

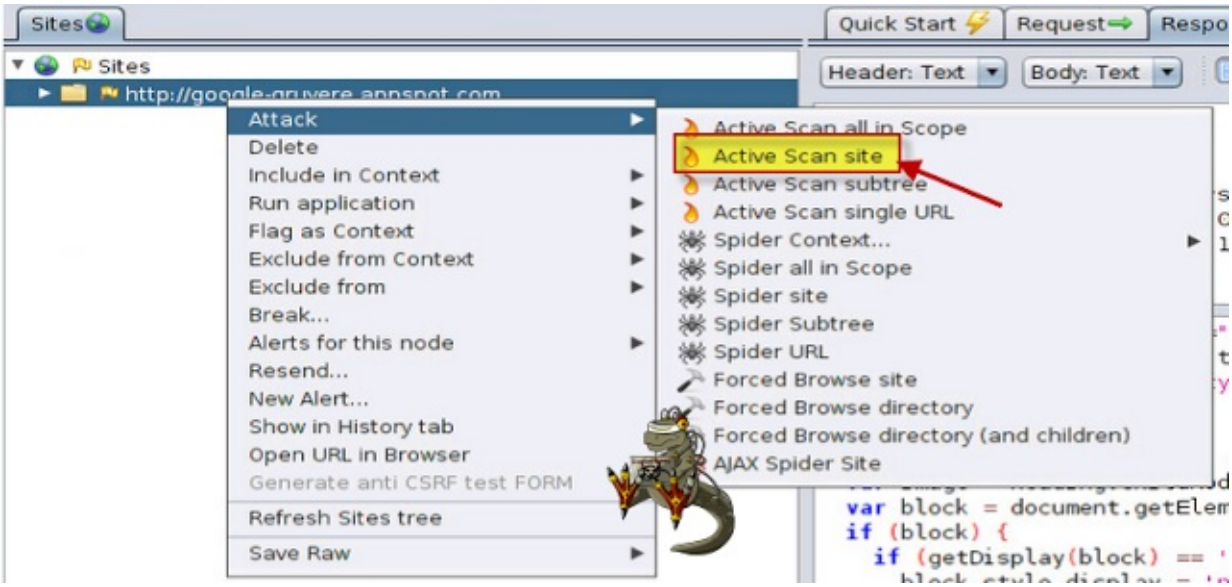
Escaneo Activo con Zap

Para llevar a cabo un scanning tenemos dos maneras, pero antes de adentrarnos en ellas veamos sus reglas y parámetros.

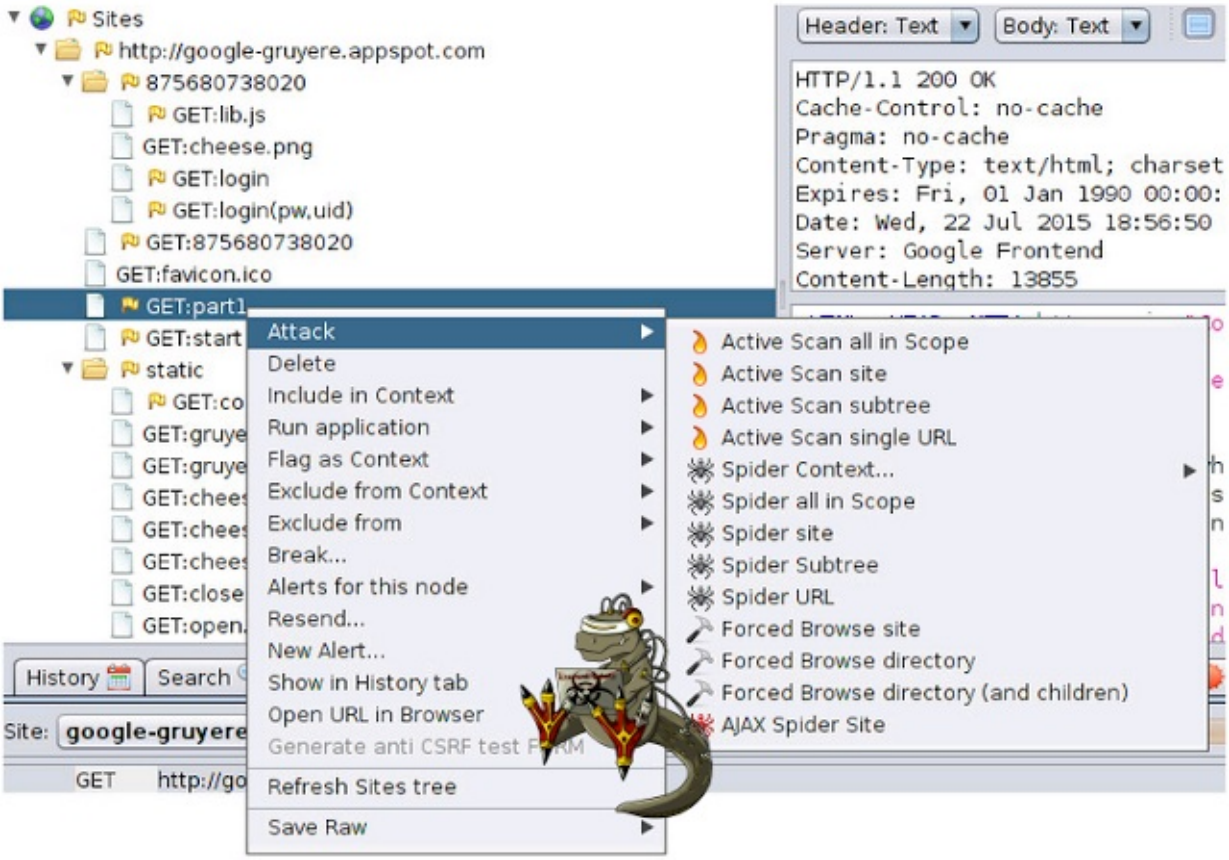
Si vamos a *tools - options - active scan* es posible configurar cantidad de host concurrentes, hilos concurrentes por host, tiempo de espera en milisegundos durante el scanning, etc.



Para inicializar el scanning activo sobre toda la url que hemos visitado durante el escaneo pasivo, sólo debemos seleccionarla, hacer click secundario y luego elegir *attack - active scan site*



Podemos manejar de manera más granular el target a escaner, para esto debemos seleccionar el subdirectorio o parámetro y elegir active scan subtree.



Por último, si no estamos interesados en identificar el targuet con un escaneo pasivo focalizando en lo que nos interesa con un escaneo activo y queremos tirarle a todo de manera cavernícola, tenemos una pestaña llamada Quick start, ponemos la url y Attack.

de ésta manera no sólo corremos un scanning activo, sino también que corre el spider, fuzzea y tira brute force (lo cual veremos detallado en las próximas secciones).

Zap Web Crawling

Continuando con las funcionalidades de la tool Zed Attack Proxy, que por cierto recienliberado la versión 2.4.1 y pueden actualizar con ctrl+u o "*ayuda - comprobar actualizaciones*" si es que al abrirlo no les ha ofrecido actualizar,veremos como hacerlo en el acápite respectivo por si quieren tener la ultima versión que esperan para ver primero ello?.

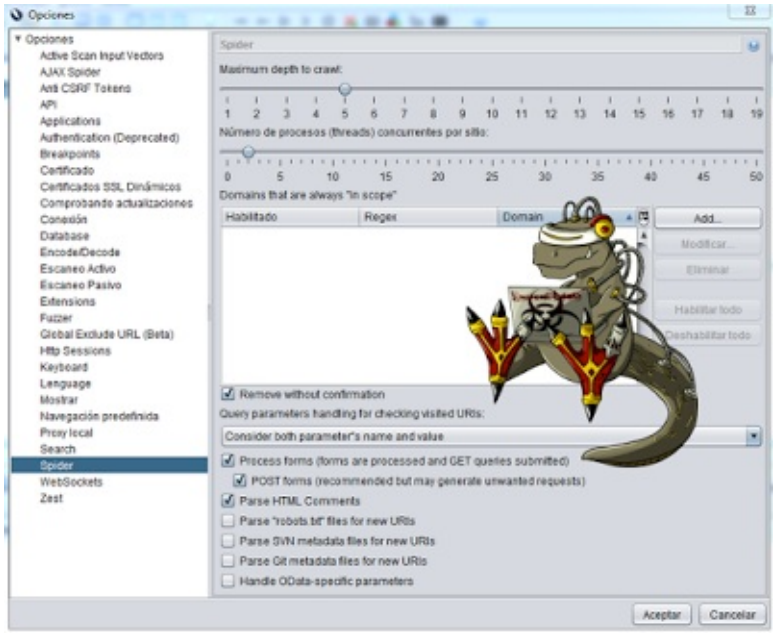
En este capítulo veremos las características de spidering que presenta y sus diferencias.

Que es Crawling o Spidering

Un crawling o spidering es una herramienta, o en éste caso una funcionalidad de Zap, que sirve para identificar los enlaces existentes en un target, de ésta manera llegamos a tener una idea de la manera en la que está compuesta el sitio a analizar e identificar posibles directorios o archivos sensibles que nos pueden ser útiles a la hora de nuestra auditoría.

La forma en la que Zap trabaja es recursiva, es decir que a medida que encuentra nuevos enlaces los va siguiendo, identificando así href, src, http-equiv o location entre otros atributos de html, get y post en lenguajes dinámicos e incluso los vínculos que están escondidos de los bots de indexación en el robots.txt. Esto nos da la posibilidad de hacer el crawling bastante granular, ya que todo es seteable desde las opciones de configuración.

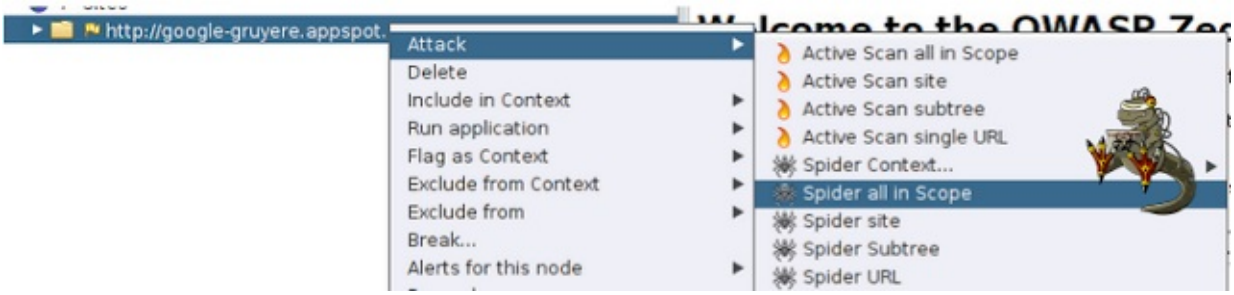
Si vamos a *"herramientas - Opciones - Spider"*, observamos las características configurables como la cantidad de procesos concurrentes, habilitar/deshabilitar el crawl en la metadata de archivos o en los comentarios del sitio, entre otras opciones.



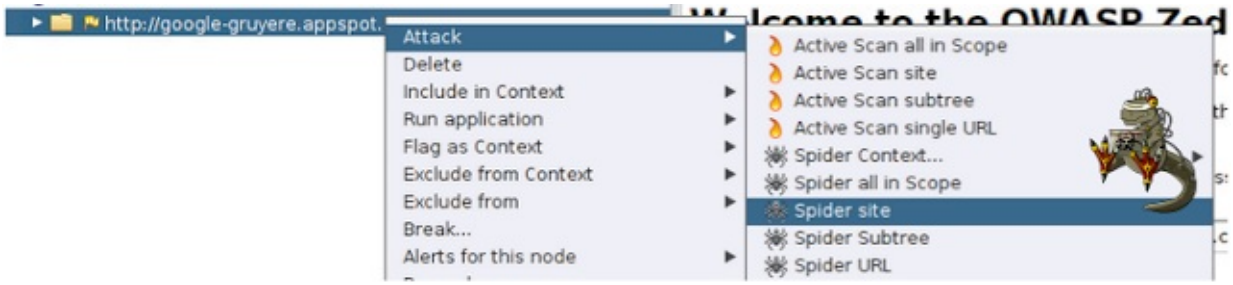
Además de tener la posibilidad de configurar cómo va a ser el spidering, también podemos configurar dónde queremos hacerlo. Partiendo desde una Url target, tenemos cinco métodos para setear el alcance:

- **Spider Context:** Analizará los enlaces seleccionados dentro del contexto seleccionado, en éste caso, la única opción es 1.

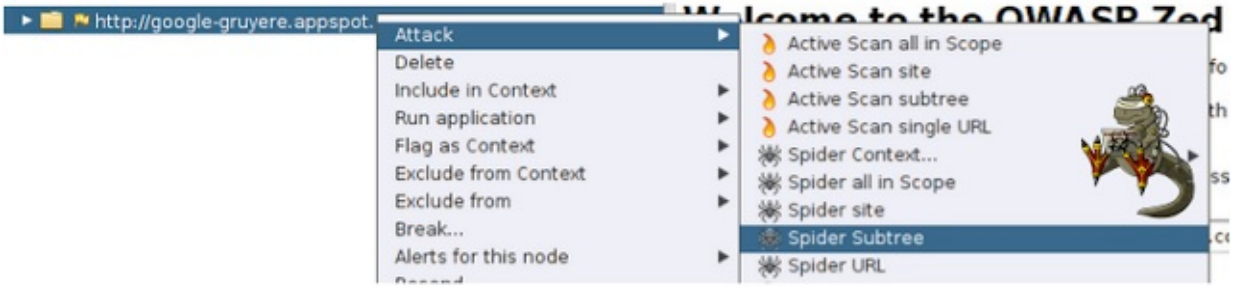
- **Spider all in Scope:** Analizará lo que le hayamos definido como alcance, en éste caso el alcance es la url principal.



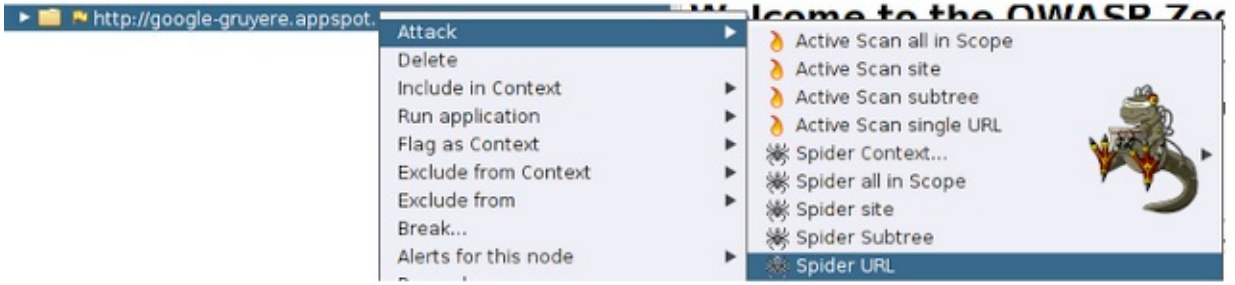
- **Spider Site:** Hará un crawling por todas los enlaces ya descubiertos en el sitio



- **Spider Subtree:** Identificará directorios y subdirectorios dentro del nodo seleccionado.



- **Spider URL:** Analiza todas las urls identificadas y las que se generan a partir de ésta.



Para comprender mejor las diferencias entre métodos de crawling, les recomiendo ingresar a un sitio con Zap como

scanner pasivo (como vimos en el capítulo anterior) y lanzar los cinco métodos diferentes identificando que nos devuelve la ventana de Spider en el panel inferior.

Fuzzing con Zap

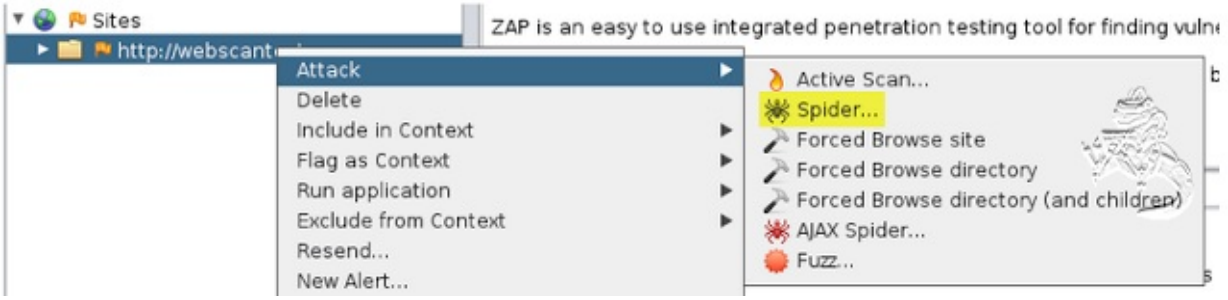
Zap, posee una gran cantidad de funcionalidades, en esta sección veremos la forma de realizar un fuzzing sobre una aplicación web para identificar inyecciones de tipo sql.

¿Que es el Fuzzing?

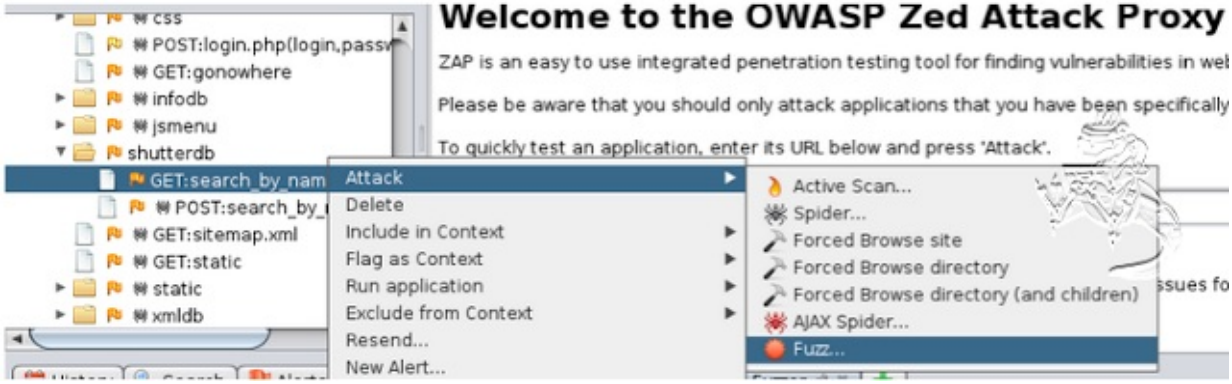
El fuzzing, es una técnica mediante la cual se puede comprobar la forma en la que responde, en éste caso una web application, ante el ingreso de datos aleatorios o secuenciales para para identificar directorios o archivos, detectar vulnerabilidades de inyección de código e incluso para realizar validaciones por fuerza bruta.

En el siguiente ejemplo, veremos la manera de identificar una inyección sql a través de un fuzzing a una aplicación web.

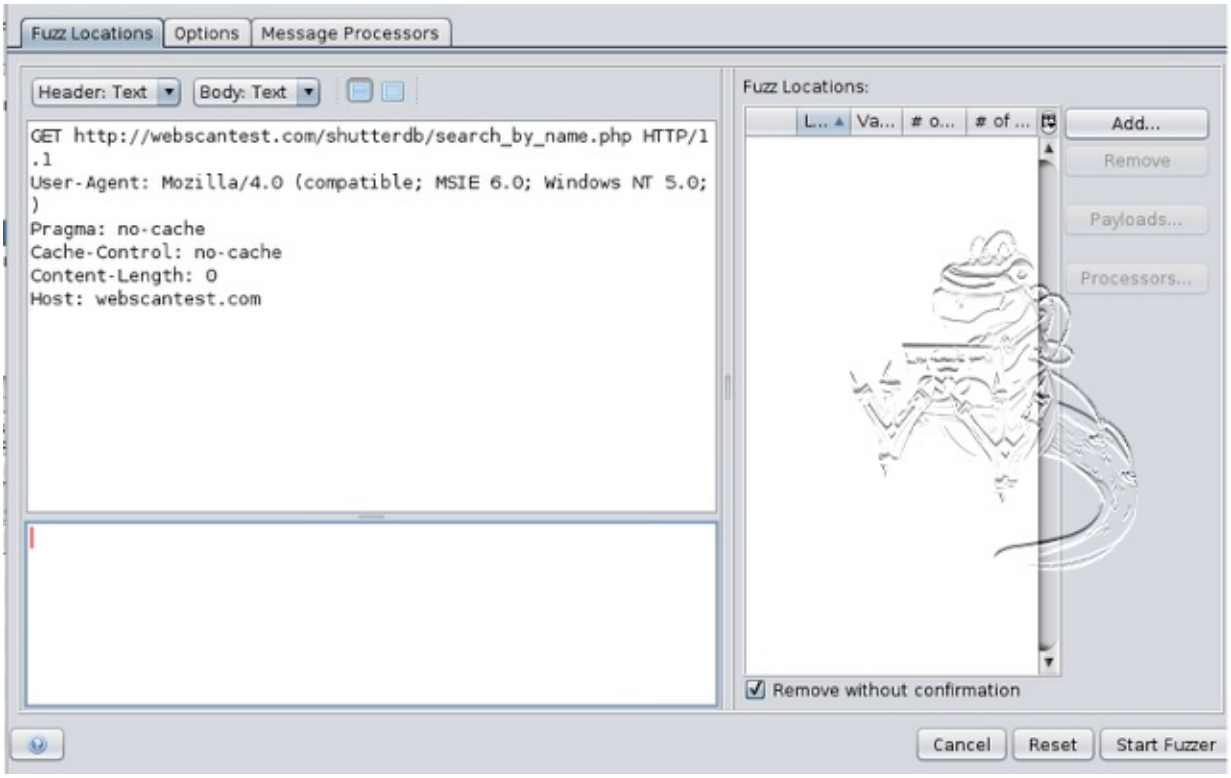
Para el siguiente ejemplo utilizaremos el target http://webscantest.com, en éste caso, he accedido mediante proxy interceptor, una vez que tenemos la url dentro de *'Sites'* click derecho *'Attack'* y luego *'Spider'*.



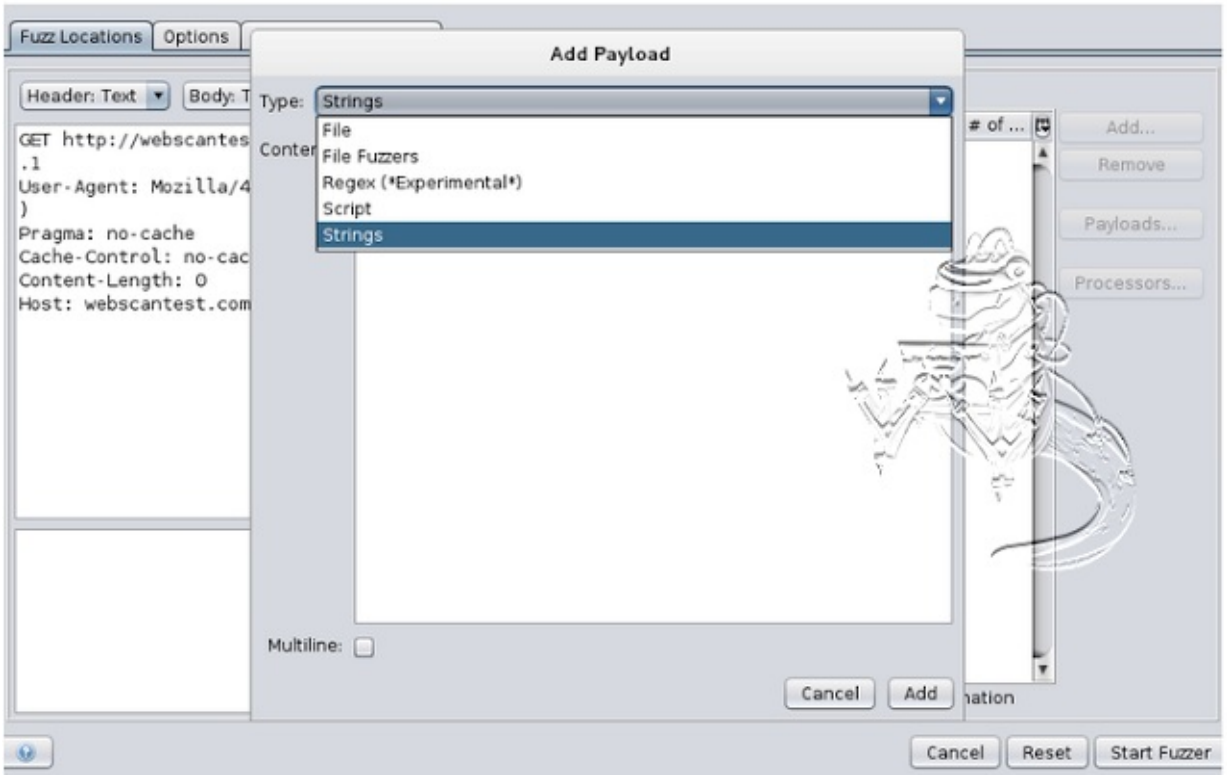
Una vez listo el spidering, vamos a buscar el directorio *'shutterdb'*, éste contiene un archivo php llamado *'search_by_name.php'* .



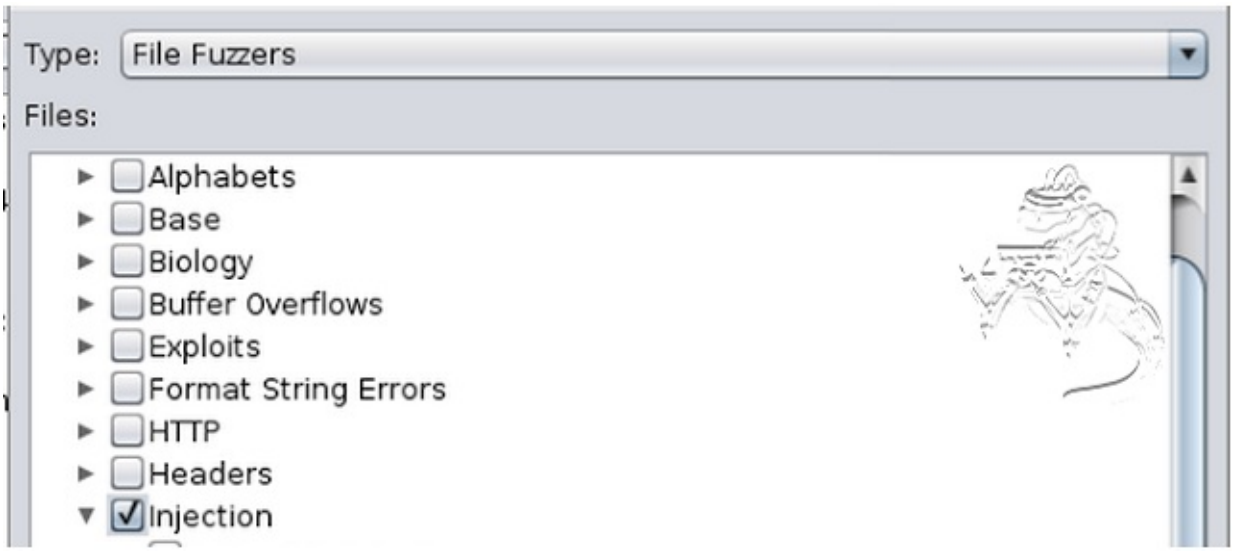
Aquí empieza lo divertido, ya que es donde podemos configurar nuestro fuzzer a gusto. Se abrirá una ventana donde veremos el header del sitio en la parte superior y en la parte inferior tendremos que setear nuestra tool.



Tras clickear en la zona inferior vamos a *'add'*, aqui es donde configuraremos los payloads o cargas a probar sobre el sitio. Debemos seleccionar nuevamente *'add'* y tendremos los tipos de cargas para lanzar nuestro fuzzer.

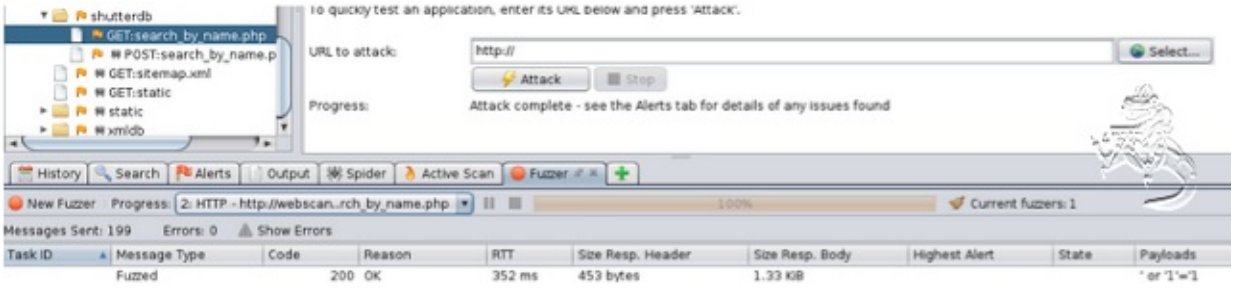


Vamos a *'File Fuzzers'* seleccionamos *'jbrofuzz'* que es otra tool de OWASP y dentro de sus opciones elegimos *'Injection'*.

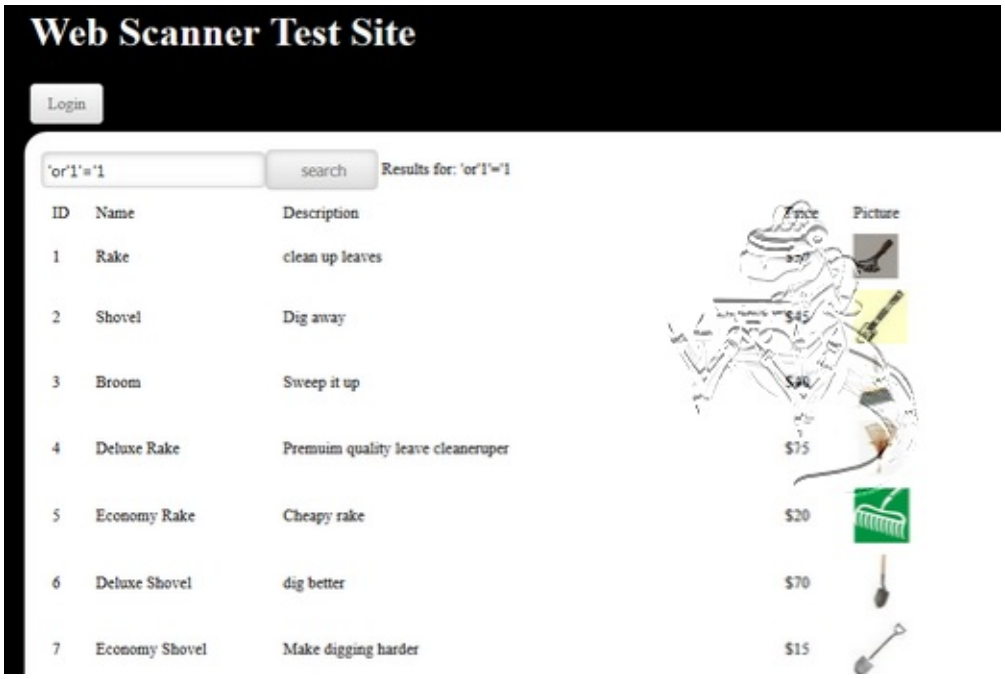


Ahora aceptamos, damos ok y demás hasta que encontramos el *'Start Fuzzing'*.

Comenzará a comprobar uno a uno los payloads imprimiendo en pantalla la respuesta de cada uno de ellos sobre el target.



Como se observa, la carga *'or'1'='1* a respondido un ok. Si vamos al sitio web y colocamos esa secuencia de caracteres podremos ver que nos devuelve todo el contenido de la db.



Me gustaría aclarar que el ejemplo que hemos visto es el paso a paso de sólo una explotación mediante fuzzing, el descubrimiento de directorios por ejemplo se puede realizar casi de la misma manera, solo basta con configurar las

opciones del fuzzer.

La intención de éste capítulo es mostrar la granularidad con la cual se puede configurar Zap, ahora bien, si colocas la el nombre del sitio en *'URL to Attack'* y das en *'Attack'* también hará un fuzzing, la diferencia está en la prolijidad, una cosa es atacar todo para que salten todas las alarmas y otra muy diferente es sólo atacar al punto débil.

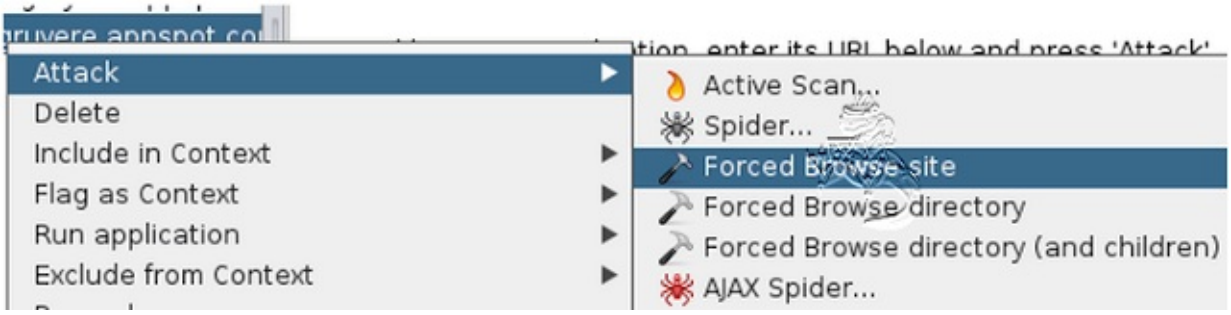
Zap Forced Browse

El Forced Browse es un tipo de ataque para forzar la navegación dentro de un dominio con el fin de identificar recursos que no son accesibles desde una referencia (*eso lo hace un crawling*) pero aun están en algun directorio dentro del web server.

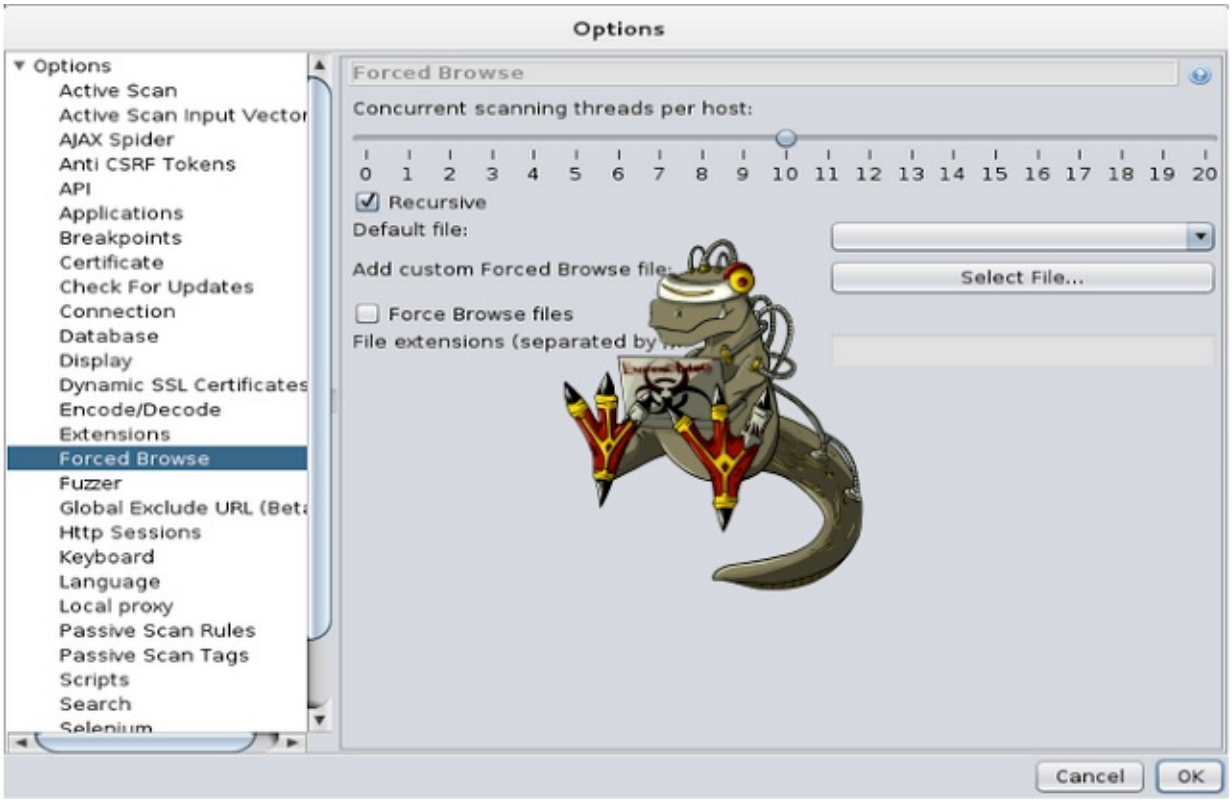
Ya que tenemos claro el concepto, analicemos su puesta en marcha, veremos que **bruteforcing** sobre la navegación viene de la mano con el capítulo anterior y el mecanismo es bastante similar.

Las opciones de Forced Browse que nos facilita la herramienta son las siguientes:

- **Forced Browse Site:** es utilizado para la identificación de contenido no vinculados en el directorio del dominio, se ejecuta por defecto o seteando nuestro propio diccionario.
- **Forced Browse Directory:** cumple la misma función que el anterior, la diferencia es que identifica sobre un directorio y no sobre todo el dominio.
- **Forced Browse Directory (and children):** Igual al anterior, sólo que además también identifica el contenido de los subdirectorios.

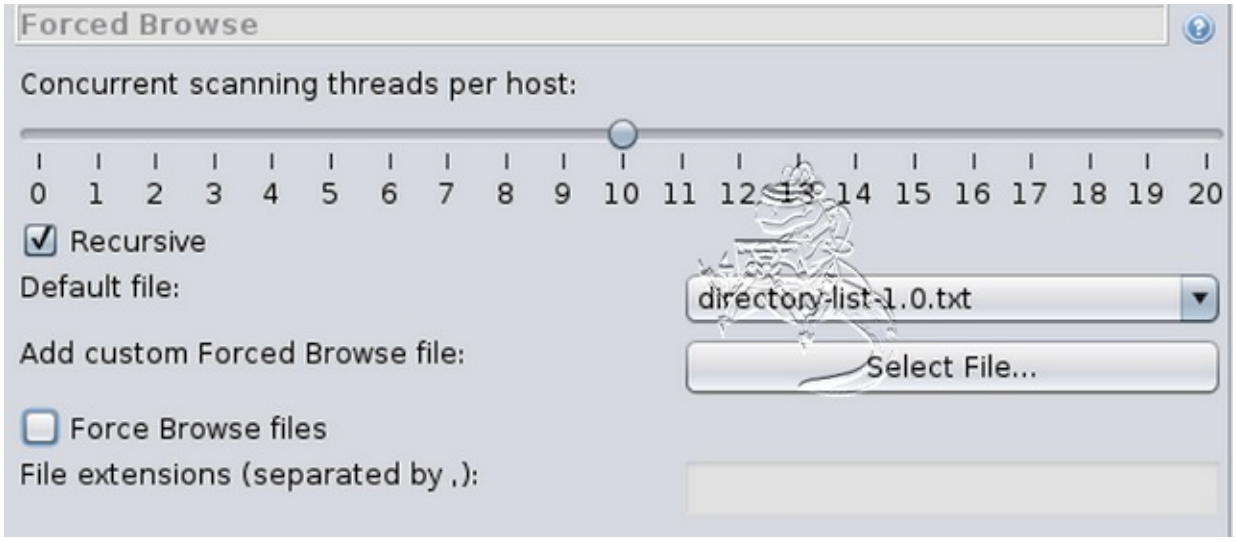


Para definir las opciones de configuración y setear nuestro propio diccionario de Fuerza Bruta en la herramienta debemos ir a las opciones, para llegar podemos ir por dos caminos *“Ctrl+Alt+o”* o en el menú superior *“Tools -> Options”*



Dentro de las Opciones configurables de las que disponemos están:

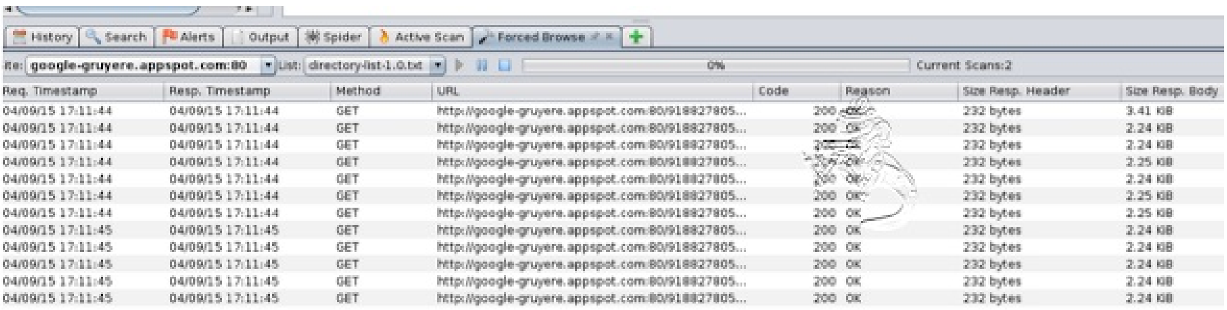
- Cantidad de threads concurrentes por host que pueden ir de 0 a 20.
- Es posible definir que sea recursivo o no.
- Archivo por defecto con el cual realizará el forced browse.
- Agregar diccionarios personalizados para el ataque.
- Incluir extensiones a buscar de manera manual.



Cuando lancemos el ataque, es viable elegir si queremos continuar utilizando la lista que hemos definido por defecto o buscar y seleccionar la que deseamos emplear.

Veamos un ejemplo práctico de lo que venimos viendo. Para Inicializar el *Forced Browse*, debemos pararnos sobre el target, click derecho y seleccionamos el tipo de *bruteforcing* que deseamos. En éste caso ejecutaremos *“Forced Browse Directory”* con el diccionario que trae por defecto la herramienta

En el panel inferior, veremos como comienza a desplegarse las peticiones que hace sobre el directorio seleccionado y la respuesta de cada request.



De ésta manera identificamos por ejemplo, información sensible, backdoors shell en php con nombres comunes como c99.php o DAws.php, etc que generalmente no tienen ninguna referencia o hipervínculo desde el dominio pero si están presentes dentro del webserver.

Zest Script sobre Zap

Antes de comenzar a ver la forma de realizar scripts en Zap, veremos de qué se trata Zest, ya que si no entendemos bien la base, es complicado construir hacia arriba.

¿Que es Zest?

Zest, es un lenguaje de programación especializado en el scripting creado por el equipo de desarrolladores de seguridad de Mozilla orientada a tools de seguridad. Si quieres investigar un poco más, acá está el sitio oficial: (**Mozilla Projects Zest**).

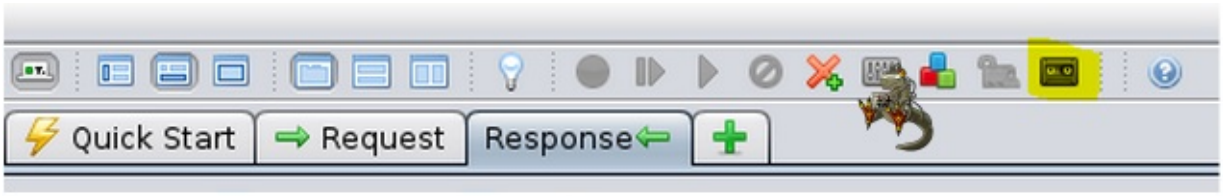
Este lenguaje de scripting está escrito en json, sin embargo, la idea es que la programación sea visual y que esté definida por la herramienta que integre Zest.

Por el momento, Zap es la única herramienta (por lo que dice el sitio de developers de mozilla (**Zest Tools**) que integra ésta forma de scriptear.

Vale aclarar que Zest es completamente de código abierto gratuito y se puede incluir en cualquier herramienta de código abierto o cerrado, libre o comercial.

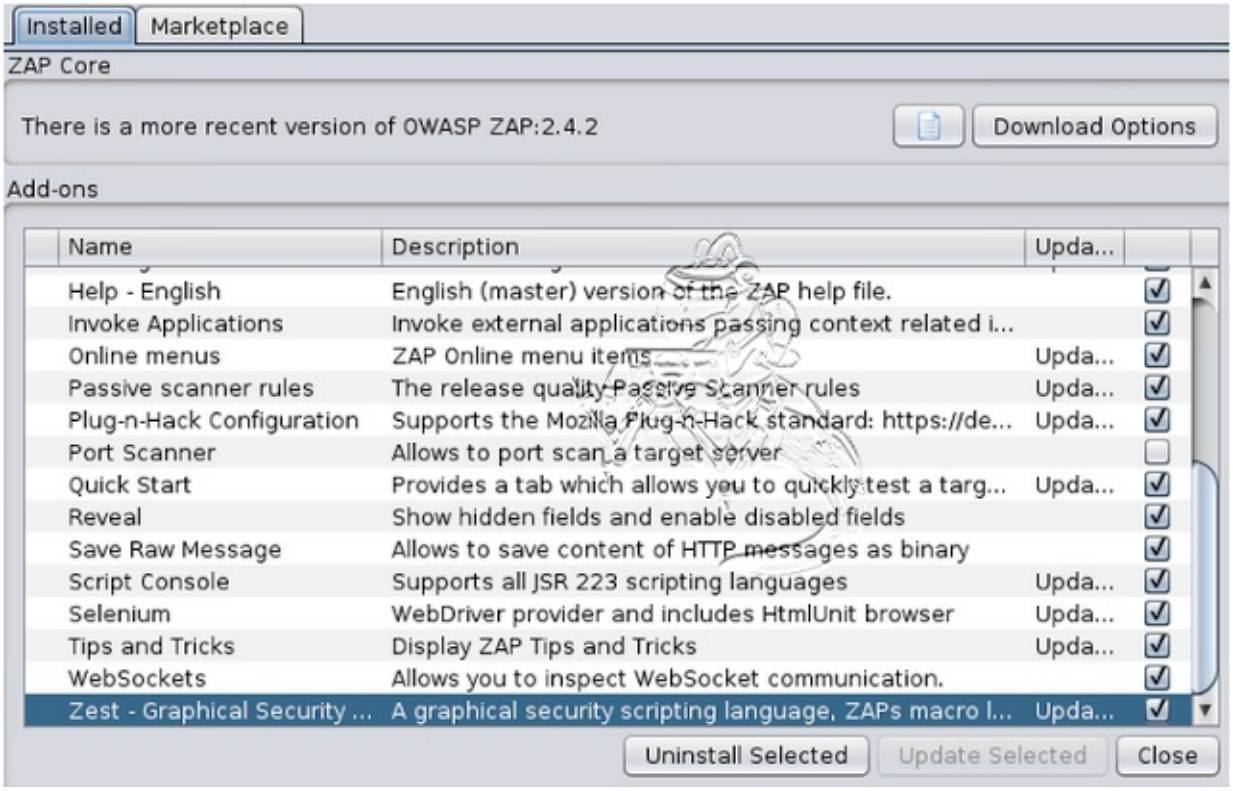
Pasos Preliminares:

Para encontrar el botón que permite grabar un nuevo script, debemos buscar en la parte superior, muy cerca de donde se encuentra el icono para customizar breakpoint http que vimos en capítulos anteriores.



En caso de que no figure ahí hay que importarlo de la siguiente manera:

1. *Ctrl+u* (Atajo para *'Check for update'* que también se puede encontrar en la pestaña *'Help'*), ésto abre el *'Manage Add-ons'*.
2. Seleccionar la pestaña *'Marketplace'* y buscar *'Zest - Graphical Security'*.
3. Una vez seleccionado vamos abajo a *'Install Selected'*.
4. Para comprobar, elegimos la pestaña *'Installed'* y debe aparecer *'Zest - Graphical Security'*, como vemos en la imagen:



Es recomendable reiniciar Zap una vez realizada la instalación del Add-on para evitar posibles inconvenientes.

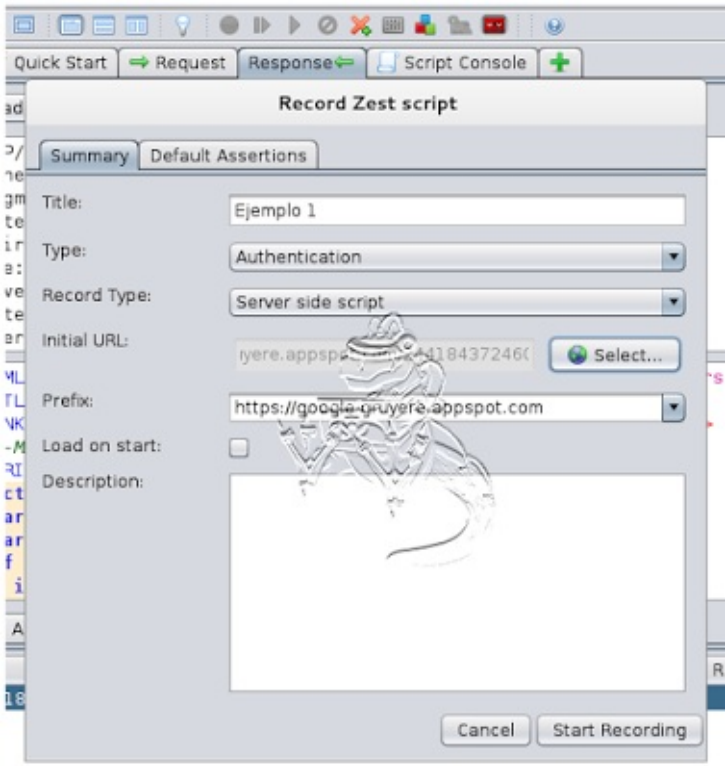
Grabación de Scripts modo visual:

Para realizar las pruebas que llevaremos a cabo, hay que tener a Zap como proxy de intercepción (lo hemos visto en capítulos anteriores).

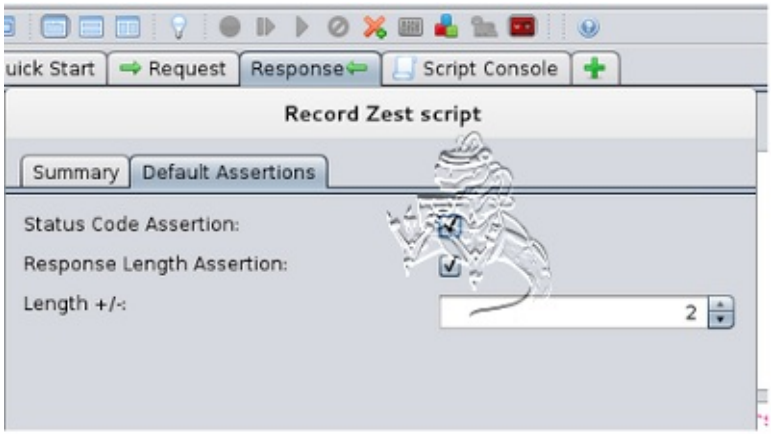
Ejemplo 1: Al hacer click en el botón para iniciar la grabación del Zest script se nos abrirá una ventana con dos pestañas, la primera, *'Summary'* nos permite:

- Setear el nombre del script.
- Tipo (stand alone o authentication)
- Tipo de grabación.
- Url inicial.
- Prefijo, es decir sobre que path raiz.
- Opción para cargar el script al inicializar la herramienta
- Descripción

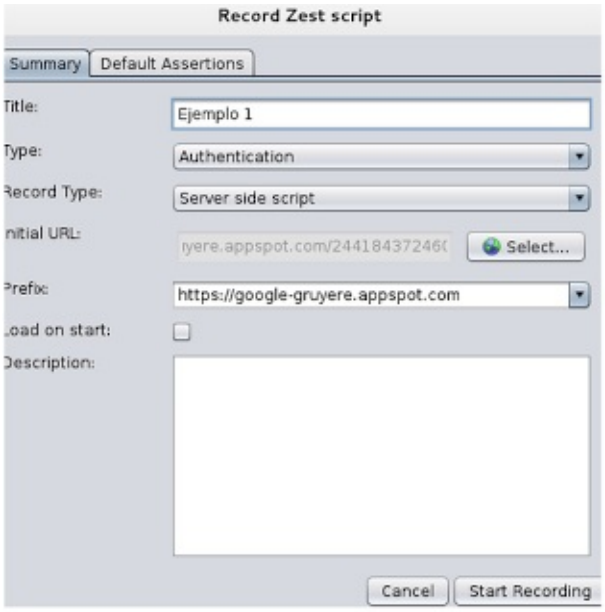
Para este ejemplo hemos elegido el nombre *'Ejemplo 1'*, el script será de *'Authentication'*, grabará del lado del servidor, la url sobre la cual correrá es la de login de google gruyere y el *'Prefix'* es el index de gruyere. No definimos descripción y tampoco que se cargue al iniciar Zap. Así nos quedó:

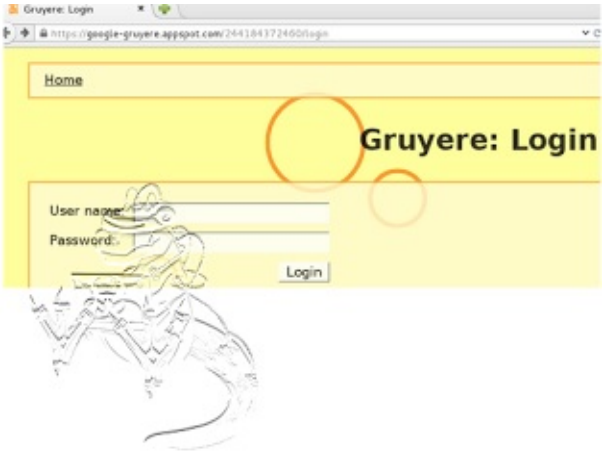


La otra pestaña, *'Default Assertions'* trae configuración por defecto que no conviene modificarlas

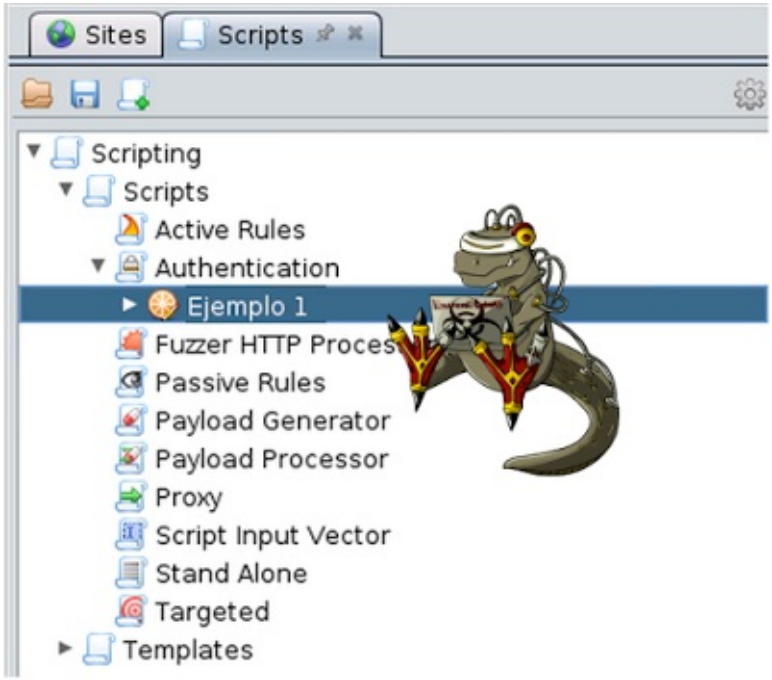


Una vez seteados los parámetros que deseamos, hacemos click en *'Start Recording'*, vamos al panel de logueo de gruyere, iniciamos sesión o introducimos credenciales inválidas y volvemos a Zap para detener la grabación del script.

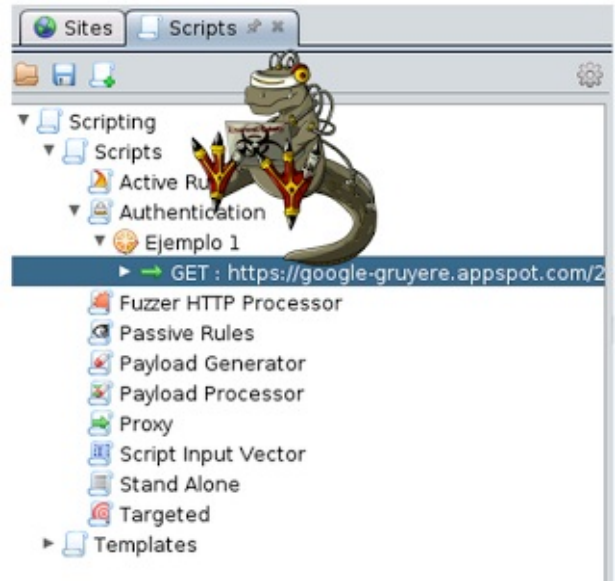




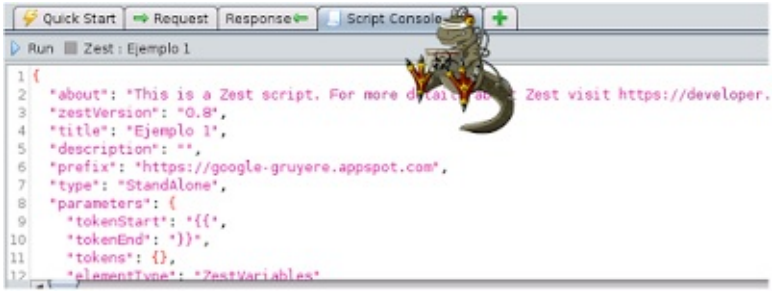
Al volver a Zap, podemos ver en la ventana *'Scripts'* que tenemos el que acabamos de grabar ya guardado.



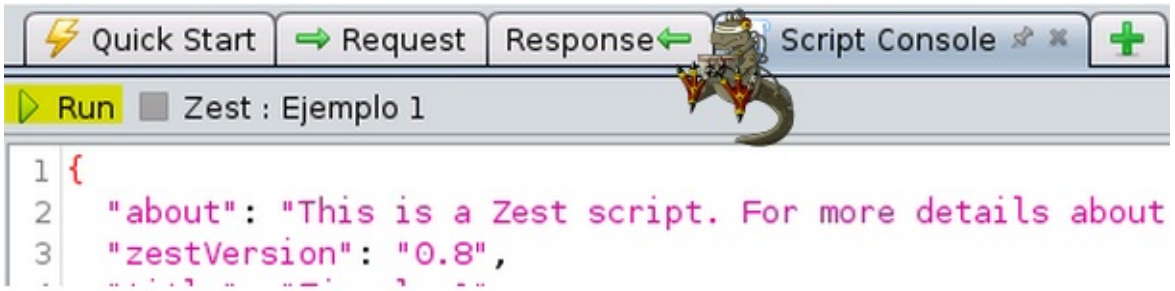
Si seleccionamos el script que hemos terminado de grabar, podemos visualizar la representación gráfica de lo que hemos grabado.



En el panel superior de la derecha, se puede visualizar la representación codificada de lo que hemos grabado, en la pestaña *'Script Console'*.



También nos apareció el botón *'Run'*, si lo presionamos el script que hemos terminado de grabar vuelve a correr.

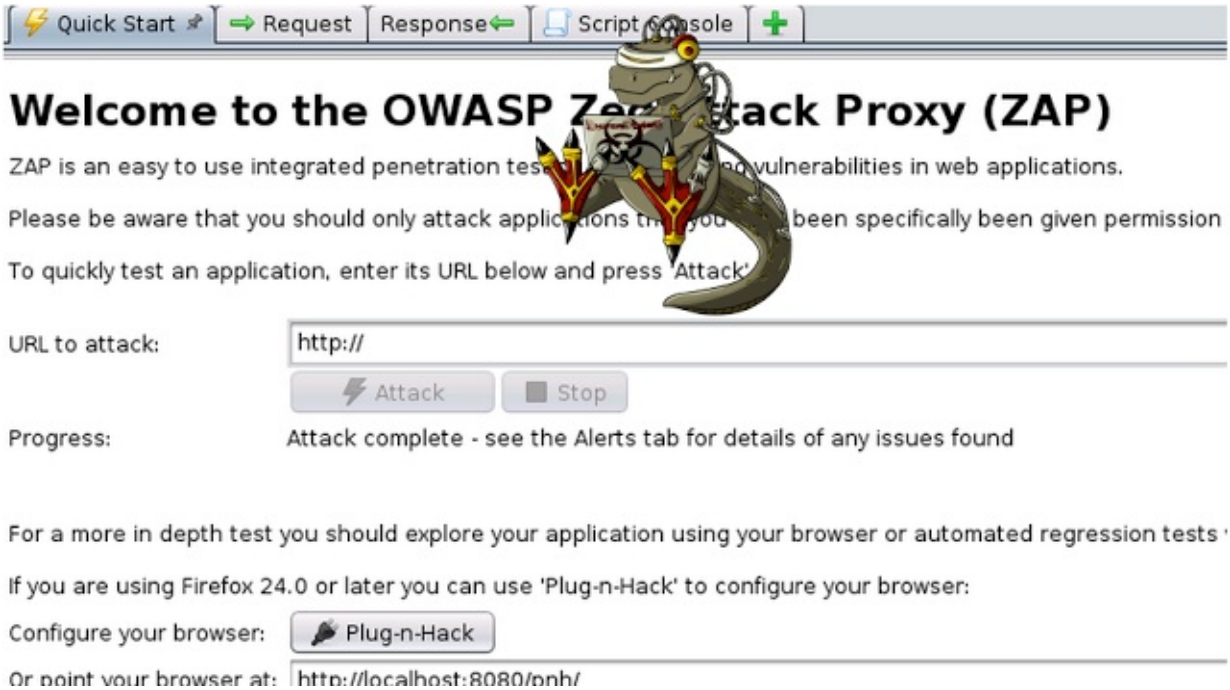


Hasta aquí, la metodología que hemos utilizado para grabar nuestro script en Zest, es muy similar a la de las macros en programas ofimáticos de procesamiento de texto o planillas de cálculo. Para resumir y dejar en claro:

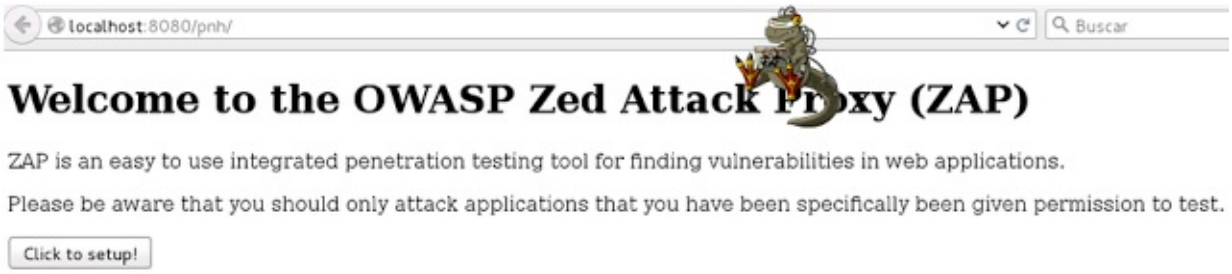
1. Vamos al botón para iniciar la grabación.
2. Realizamos el proceso que queremos dejar guardado. Aquí podemos, por ejemplo, explotar un vulnerabilidad como un xss o un bypass al login que luego podemos reproducirlo para generar nuestro reporte .
3. Finalizamos la grabación para tener nuestro script codificado y representado gráficamente. Podemos volver a correr lo que hemos grabado simplemente dando click en *'Run'*.

Grabación de Scripts modo consola:

Para realizar ésta modalidad de generación de scripts, debemos utilizar un plugin llamado *'Plug-n-Hack'* que se encuentra en la pestaña *'Quick Start'* debajo del *'Url to Attack'*.



Si hacemos click en el botón 'Plug-n-Hack' se nos abrirá una pestaña en el browser Firefox con la url: http://localhost:8080/pnh con un botón para configurar el plugin.



Una vez que finaliza la configuración, podemos ver las modificaciones que realizó el plugin dirigiéndonos a *'Preferencias -> Avanzadas -> Configuración'*.

Para generar nuestros scripts, debemos, desde nuestro browser presionar *'Shift+F2'*. Si sólo ponemos zap veremos todas las opciones que tenemos disponibles para interactuar desde Firefox con Zap.

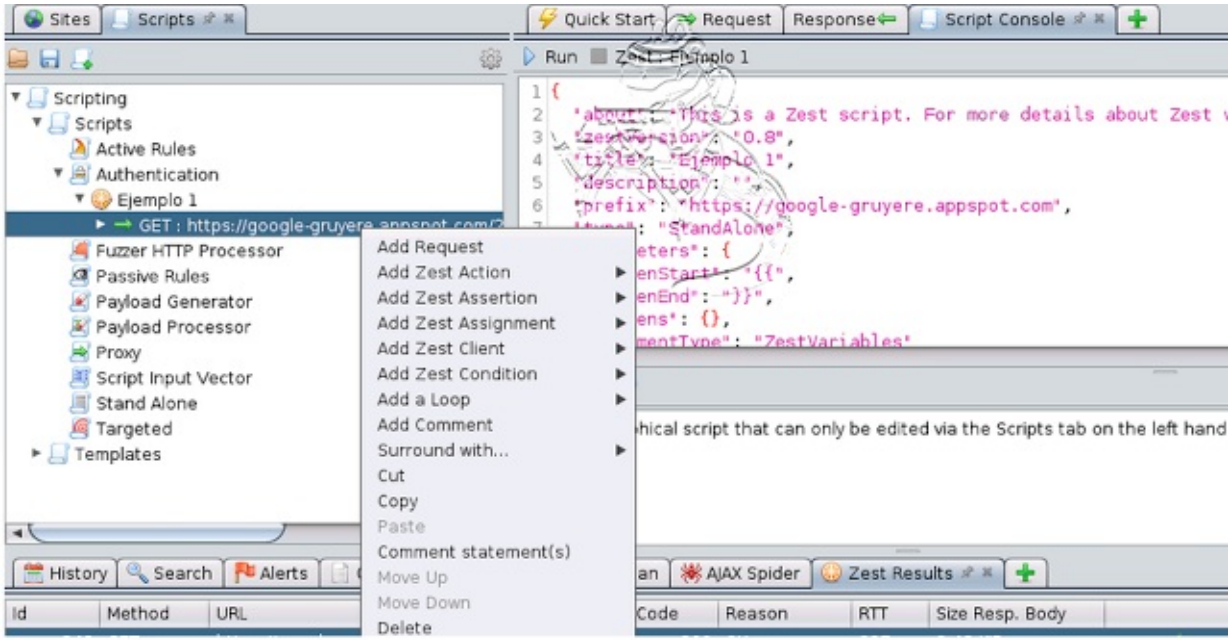
Para inicializar la grabación del script, debemos escribir *'zap record on global'*, realizamos la acción que queremos de dejar guardada en nuestro script y para finalizar escribimos en la consola de Firefox 'zap record off global'. Al vovler al Zap, nos encontraremos con los mismos paneles y visualizaciones que si grabamos un script de manera visual.

Edición de scripts:

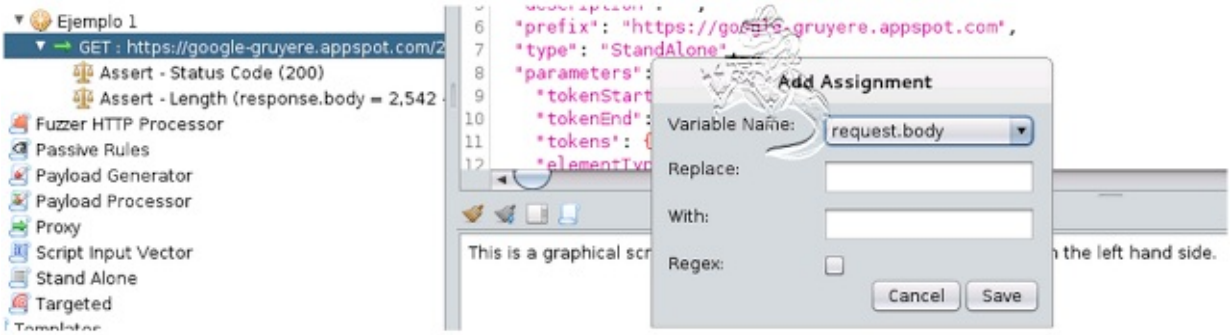
Al seleccionar un script que hemos grabado podemos realizar una gran variedad de acciones entre las cuales podemos ver:

- Agregar request.
- Agregar acciones como scanear, imprimir o dormir el script.
- Agregar afirmación length y regex.

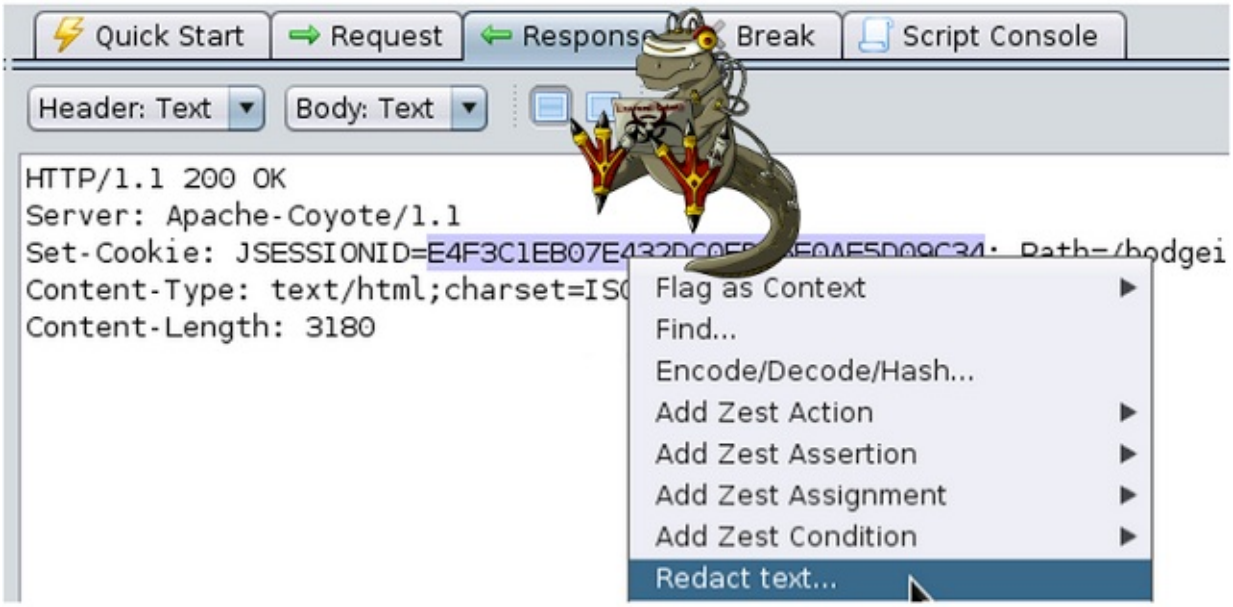
- Agregar asignaciones en variables por ejemplo.
- Agregar condiciones.



Cada una de las opciones de edición nos abrirá una ventana para editar nuestro script de manera visual.



Otra forma de editar, es ir directamente al código json seleccionar el string, elegir *'Redact text'* y modificar a gusto lo que deseamos. Por ejemplo, si obtenemos cookies de sesión de una aplicación web y hemos generado nuestro script de login con una cookie, es posible modificarla por otra y hacer click en *'Run'* para que valide el logueo con otra cookie. También podemos hacerlo con las credenciales de acceso, modificar header, variables, etc.



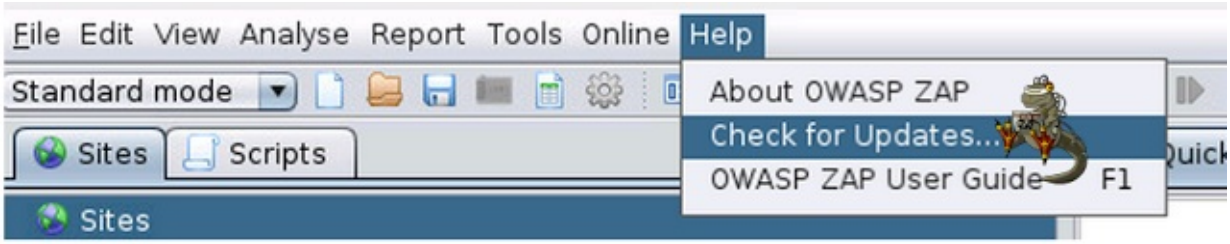
Existen muchas más opciones de modificación y personalización de los scripts, podemos agregar condiciones booleanas, iteraciones, etc, para ajustarlo a nuestras necesidades al 100%. Esa parte quedará para su investigación.

Si quieren interiorizarse más, en github tienen el código de Zest y en mozilla zest tienen un grupo para realizar las consultas o aportes sobre éste lenguaje de scripting.

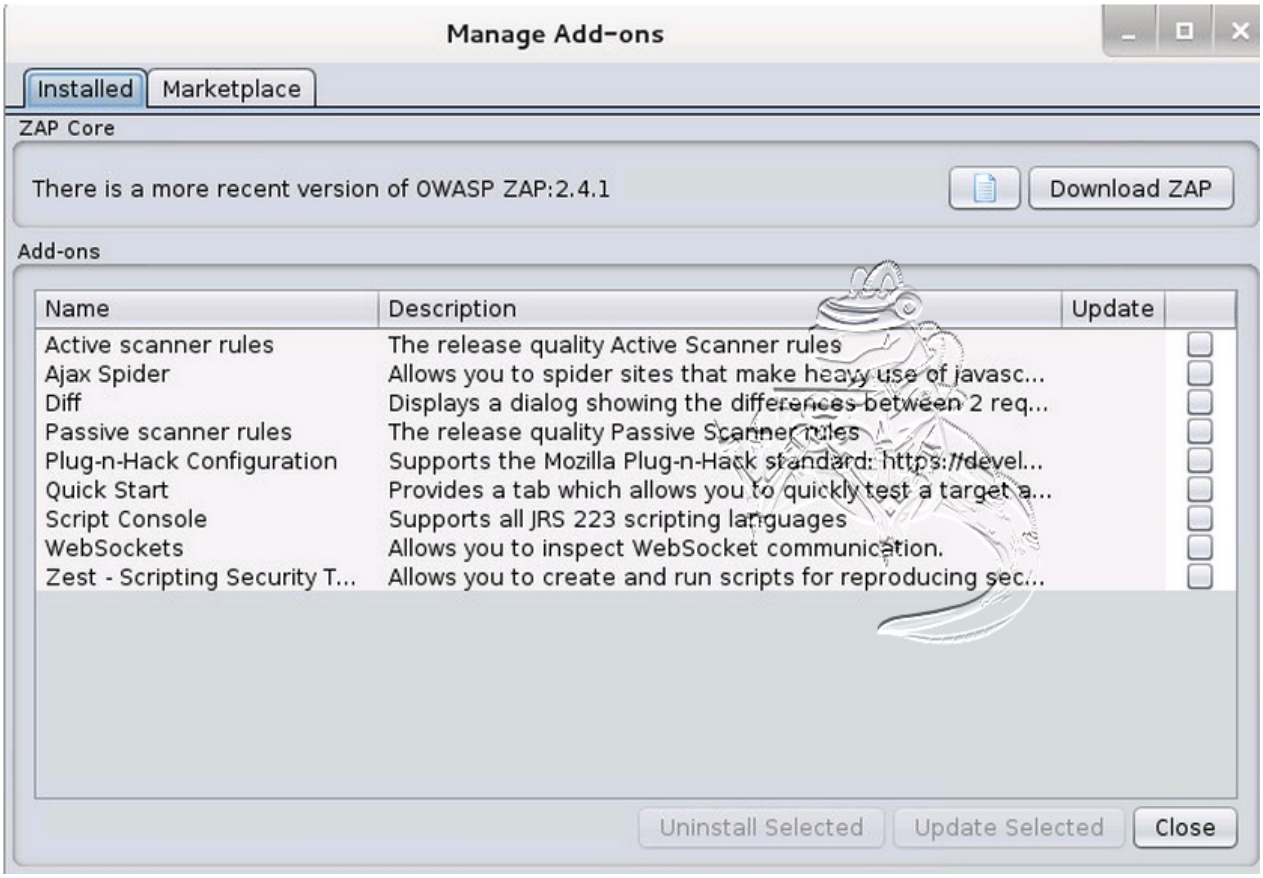
Actualización y Plugins en Zap

Aprovechando que recien se liberó la versión 2.4.2 de Zap, en este capítulo veremos la manera de actualizar, además de la instalación y uso de los plugins.

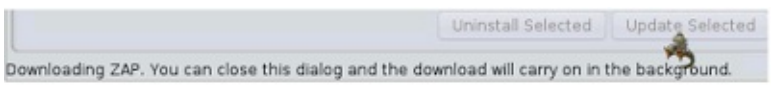
Para actualizar la herramienta, tenemos que ir a la pestaña *'Help'* y seleccionar *'Check for Updates'*.



Se abrirá el *'Manage Add-ons'* y si no tenemos la última versión, nos aparecerá un mensaje indicándonos que existe una versión más reciente.

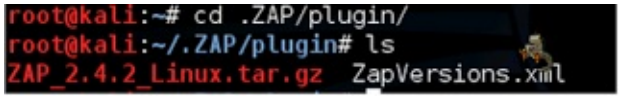


Damos click en *'Download ZAP'* y comenzará la descarga en segundo plano, como lo indica el texto en la parte inferior de la ventana.



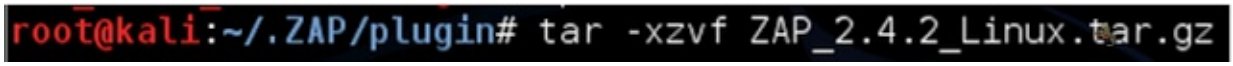
Una vez, hecho ésto, levantamos una consola y listamos con *'ls'* sobre ~.

Nos movemos al directorio *'.ZAP/plugin'* y al listar vemos que ya tenemos la versión nueva.

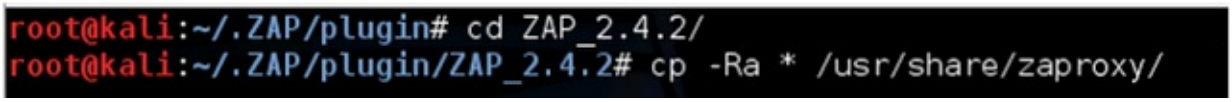


En caso de no estar el *'.tar.gz'* con la versión nueva, podemos descargarla directamente desde GITHUB o haciendo un *'wget'*.

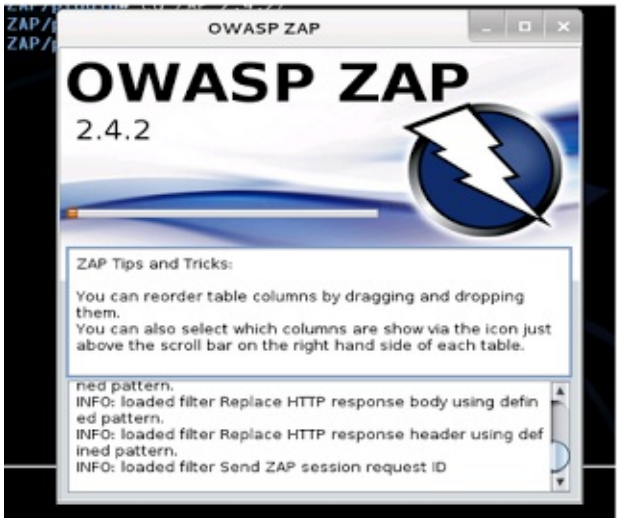
Cuando lo tengamos, debemos descomprimir el archivo compreso de la siguiente manera.



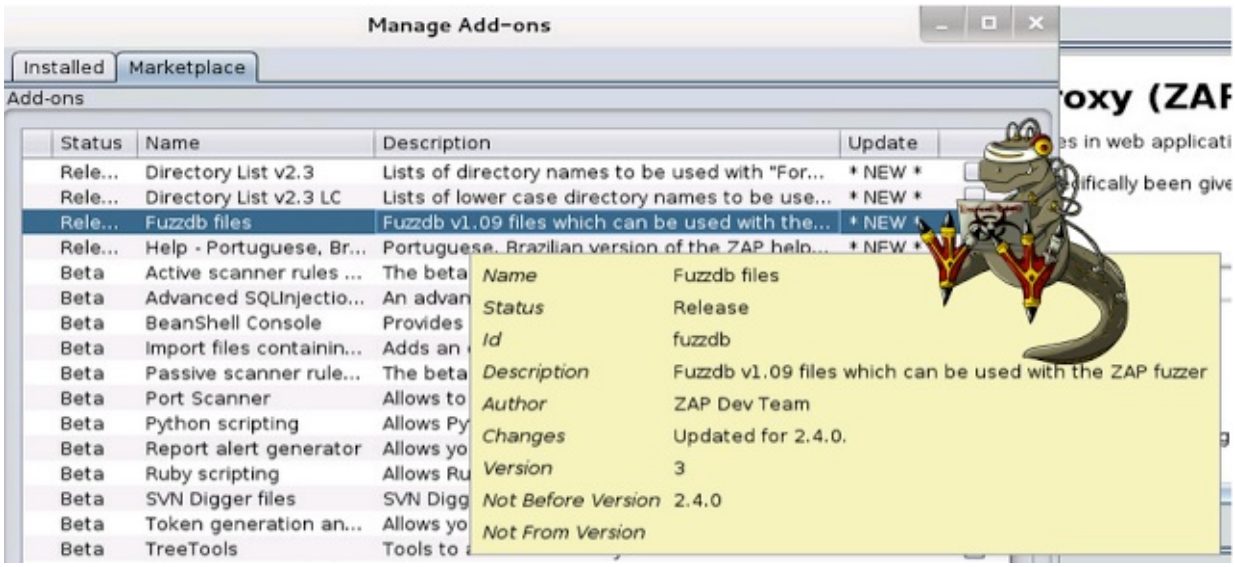
Y por último movemos toda el directorio de ZAP nuevo para pisar el viejo sobre *'/usr/share/zaproxy'*.



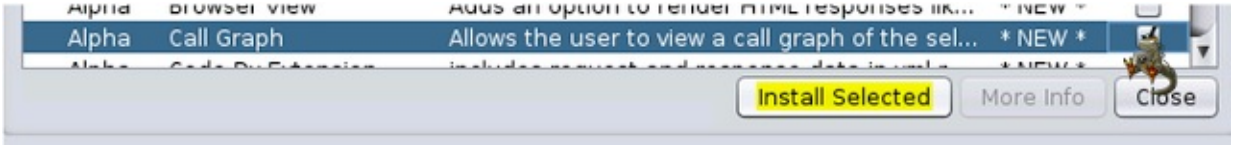
Iniciamos ZAP para corroborar:



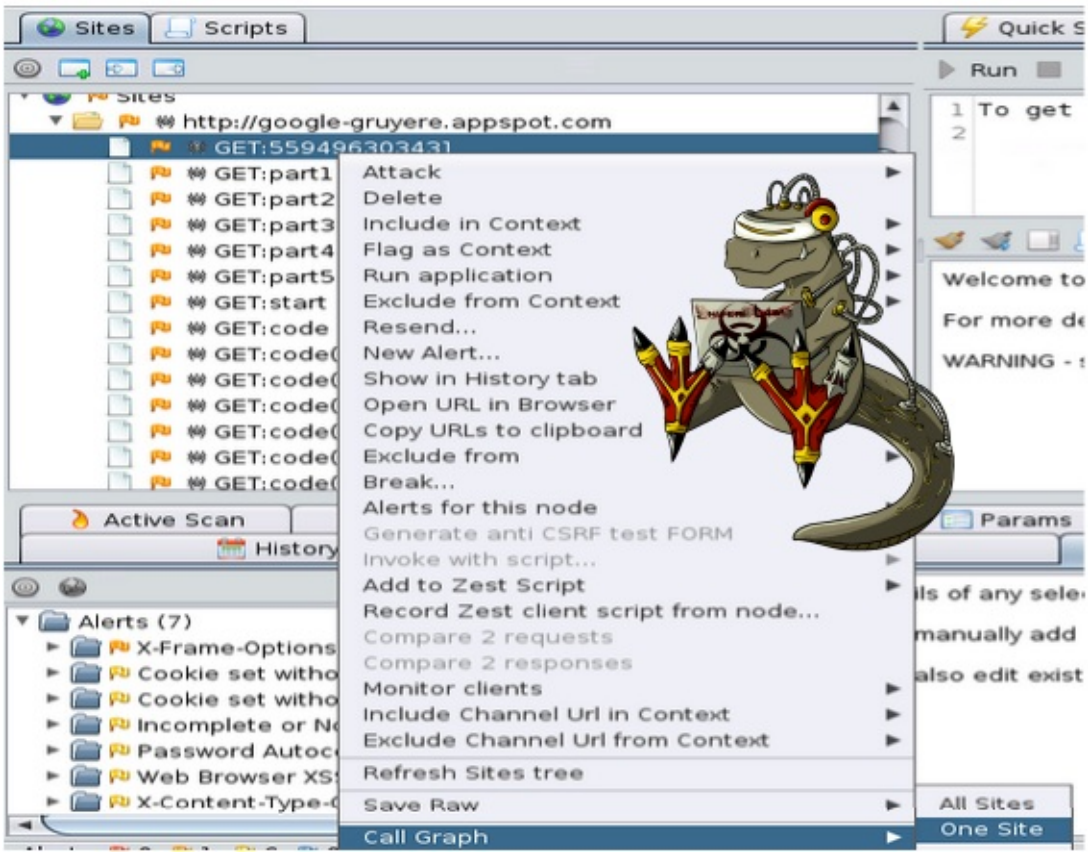
Con la última versión de ZAP, vienen varios *add-ons* interesantes y varios updates. Para visualizar los plugins debemos ir al *'Manage add-ons'* como vimos más arriba o simplemente con *'Ctrl+u'*. Si deseamos ver de queé se trata un plugin, basta con hacerle click para que nos muestre más información.



Para el siguiente ejemplo, instalaremos y veremos el funcionamiento de un plugin llamado *'Call Graph'*. Lo seleccionamos y clickleamos en *'Install Selected'*.



Una vez finalizado, vamos sobre nuestro ya conocido 'google-grouyere.appspot.com' y haciendo click secundario ya tendremos la opción del plugin que acabamos de instalar disponible.



Si seleccionamos la opción de 'All Site', veremos un diagrama de todos los directorios, subdirectorios y archivos que se ha identificado el scanner.

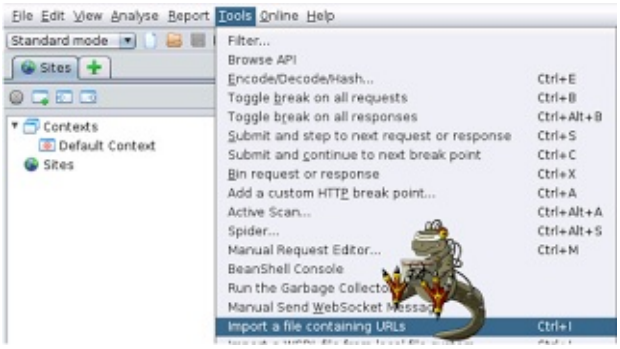
Los invito a que revisen el resto de los plugins, hay muchísimos muy interesantes como *'Log File Importer, Browser View, fuzzdb files, Python Scripting'* entre otros.

Para finalizar, quisiera agradecer a Simon Bennetts (@psiinon) y a Cristian Borghello (@SeguInfo) por el agradecimiento y difusión de las entradas del blog.

Cuestiones que nos olvidamos

En este penultimo capítulo veremos algunos recursos pequeños pero sumamente útiles de Zap que nos han quedado pendientes pero que no son los lo suficientemente extensos como para dedicarles toda una sección.

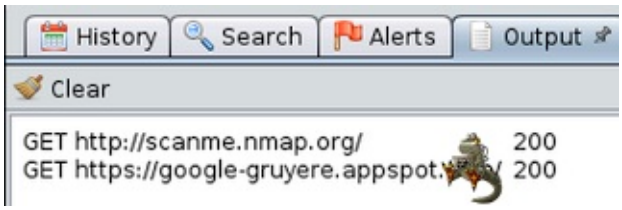
Ctrl+i: Mediante la combinación de éstas teclas, Zap abre una ventana desde la cual podemos elegir una lista de urls para poder realizar un análisis. También podemos encontrar la opción si vamos a la pestaña *'Tools'* y seleccionamos *'Import a file containing URLs'*.



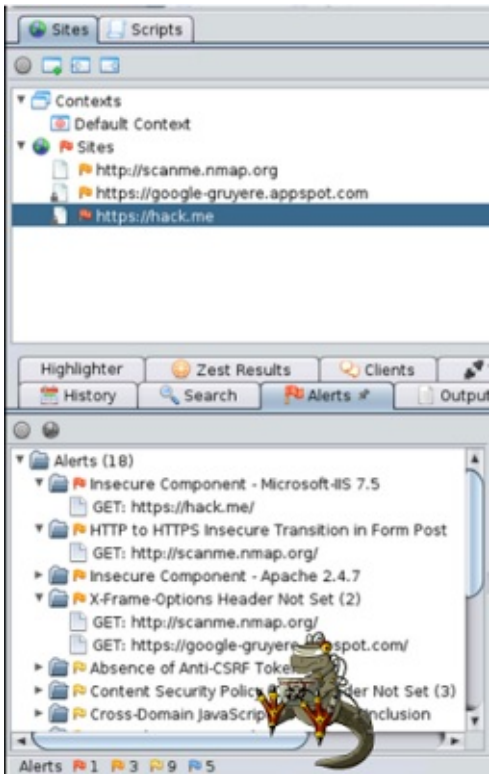
La ventana que se nos abre nos permite seleccionar el listado que tengamos armado.

Para éste ejemplo, armé un pequeño txt con tres urls que lo pueden copiar para sus prácticas.

Al importar y darle aceptar, veremos que en la pestaña *'Output'* del panel inferior comenzará realizar peticiones a los sitios de nuestra lista y mostrar las respuestas.



Una vez finalizado el scanning, en la pestaña *'Sites'* veremos los targuets analizados y en la parte inferior, si vamos a *'Alerts'* tendremos las vulnerabilidades identificadas en cada una de las urls.



Las Apis de Zap

Zap está desarrollado en Java, pero cuenta con apis para poder integrarlo a otros lenguajes de programación como Ruby, Python, PHP, entre otros. En el siguiente ejemplo veremos la forma de integrar Python a la herramienta.

Existen dos opciones para instalar la api de python en zap:

La primera es desde PyPI.

```
root@kali:~# pip install python-owasp.zap-v2.4
```

para luego desde el directorio de python.

```
root@kali:~# cd python/api
root@kali:~# python setup.py install
```

La segunda opción (que es la recomendada), es bajando la api desde Sourceforge para luego seguir con los mismos pasos del procedimiento descrito arriba.

Una vez ejecutado el install, tenemos que inicializar el daemon de zap.

```
root@kali:~# ./ZAP_2.4.2/zap.sh -daemon
```

Hasta aquí ya tenemos instalada la api y lista para utilizarla. Ejemplos de scripts que usan la api de python pueden encontrar aqui o aqui.

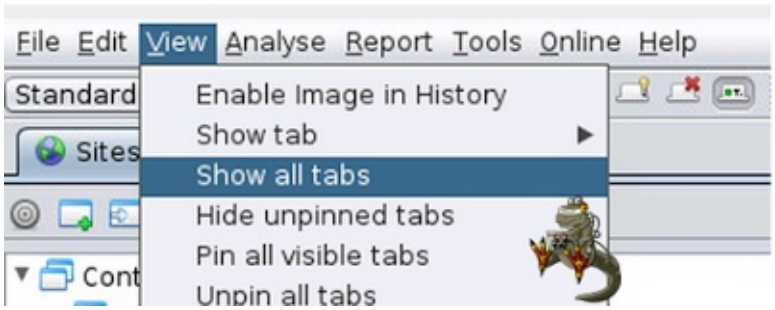
Port Discovery

- He, pero eso lo hago con nmap.
- Perfecto, pero también podés hacerlo con Zap papu!

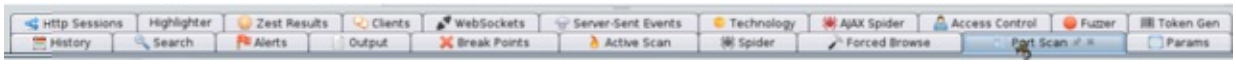
Por si alguien aún no sabe qué es un port scanner, se trata de identificar los puertos abiertos de un servidor. Generalmente los puertos abiertos (si están bien configurados) tienen un servicio corriendo, como smtp en el 25, https en el 443, o pop3 en el 110 por ejemplo.

Con Zap, podemos ver los puertos y servicios asociados sin la necesidad de abrir un nmap o la tools que usen para éste propósito. El conocer los puertos y servicios de un servidor, nos pueden orientar para saber de qué sistema operativo (SO) se trata y validar en una análisis de vulnerabilidades o pentest si algunos críticos como ftp o ssh son bypasseables fácilmente o si cuentan con las medidas de seguridad correspondientes.

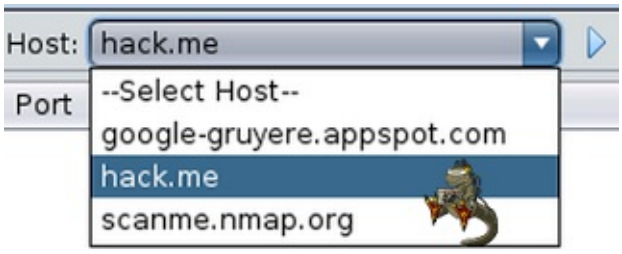
Para el siguiente ejemplo, llamemos nuevamente a listado de urls que vimos más arriba con *'Ctrl+i'*. Ahora, vamos a la pestaña *'View'* y elegimos *'Show all tabs'*.



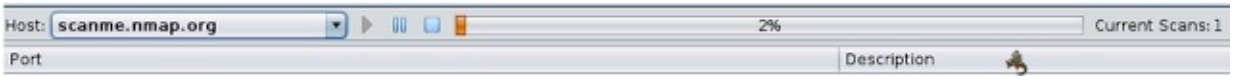
Si miramos las pestañas del panel inferior, notaremos que hay muchas más de las que trae por defecto cuando se inicializa la herramienta.



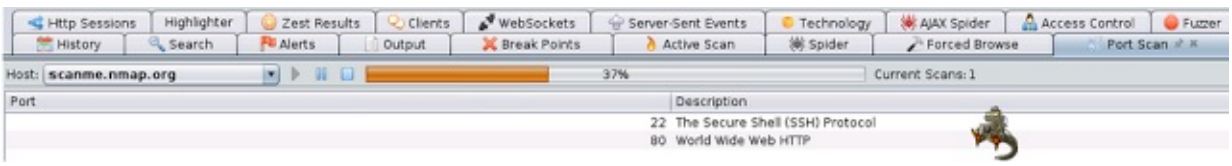
Elegimos la pestaña *'Port Scan'* y en *'Host'* el target a utilizar para la identificación de puertos abiertos.



Luego, sólo click en *'Play'* y a esperar.



Al **37%** ya encontró dos puestos abiertos con sus respectivos servicios asociados

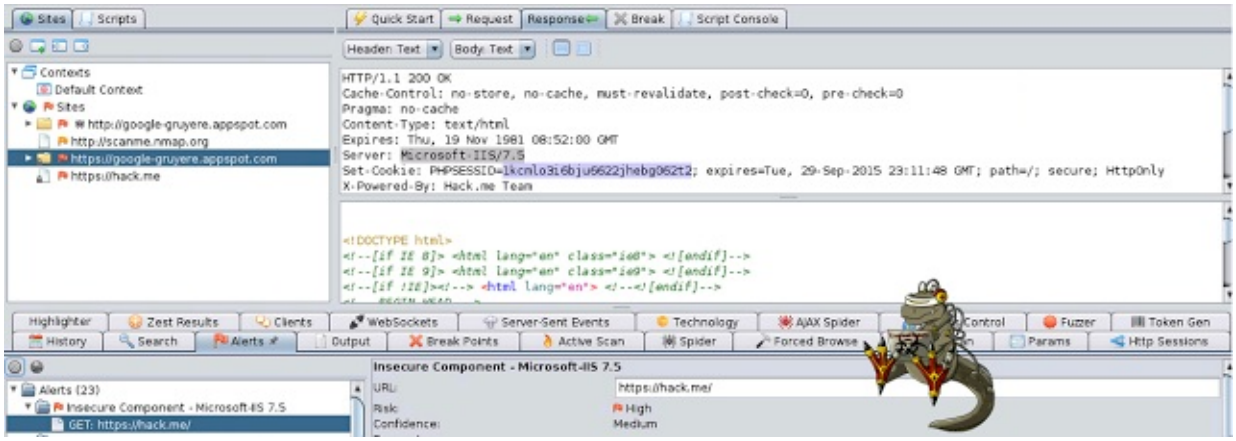


Generador de Cookies

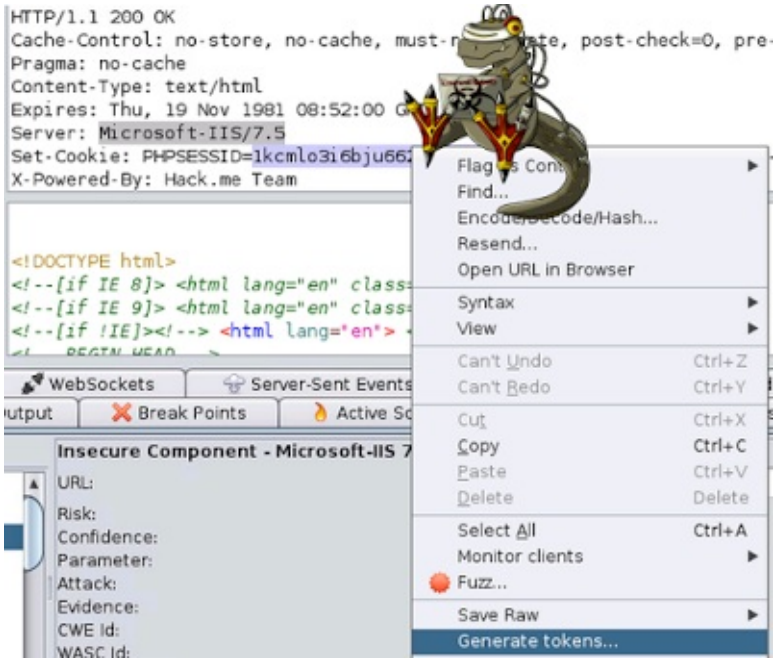
El generador de token nos sirve para validar una cookie, probar si podemos loguearnos a un sitio con una cookie inventada o utilizar alguna cookie capturada para realizar la auditoria desde adentro.

Para éste ejemplo, seleccionaremos la primer alerta que nos dió Zap al importar el listado. Nos paramos sobre *'Insecure*

Componnent' y veremos la versión del IIS de Microsoft, pero debajo encontramos el **'PHPSESSIONID'**

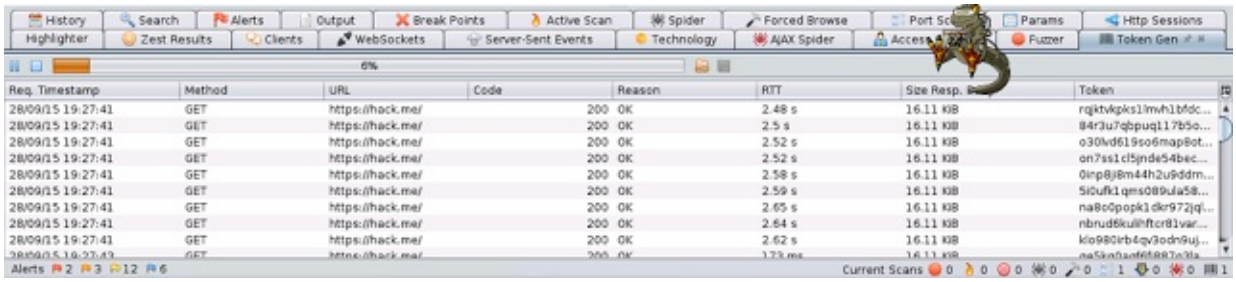


Al hacer click secundario, abajo tenemos la opción de *'Generate tokens'*.

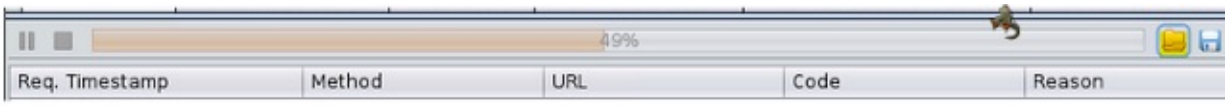


Aquí podremos setear la cantidad, el tipo y el nombre.

Vamos a *'generar'* y visualizamos los resultados.



Si vamos al icono de la carpetita, podemos incorporar las cookies que tengamos para realizar las pruebas.

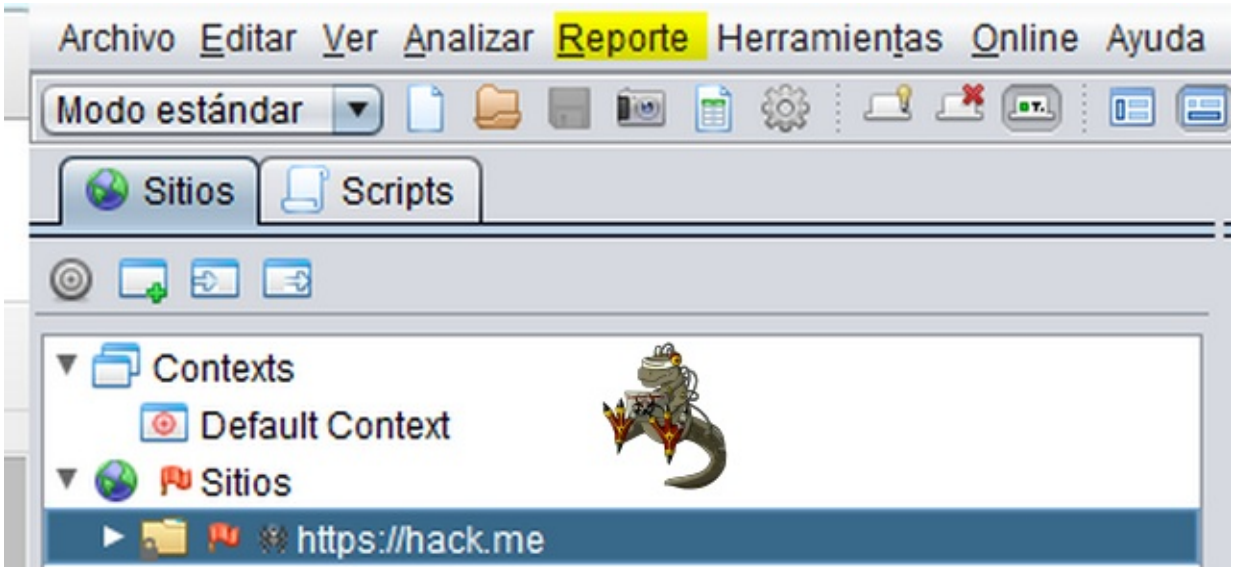


Reporting

Nos encontramos ya en la último capítulo al menos por ahora, veremos puntualmente la manera de manejar los reportes que nos permite exportar la tool.

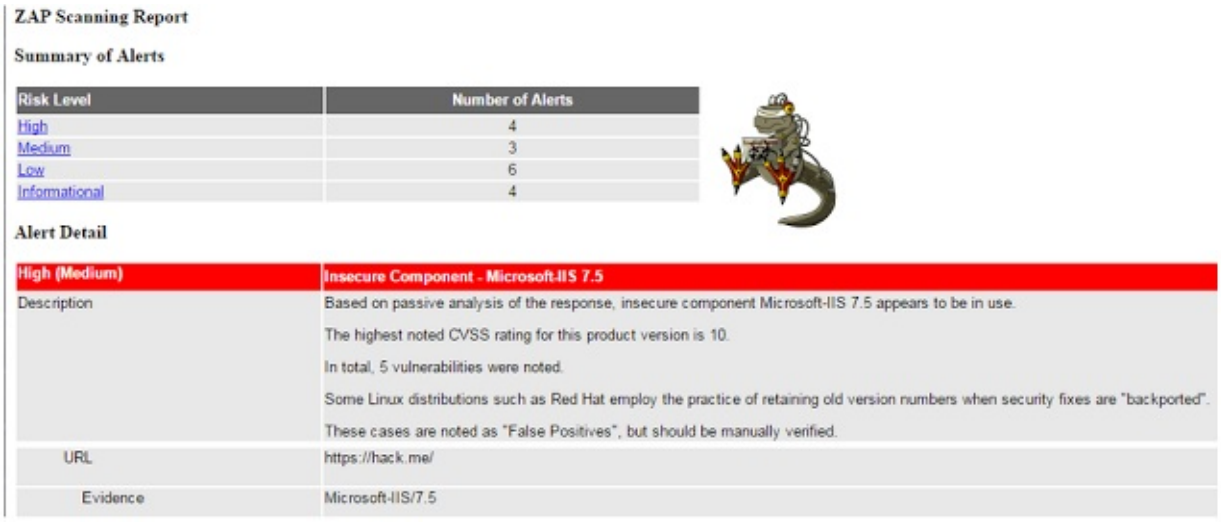
Generar un reporte.

Una vez que terminamos de analizar el sitio, podemos ir a la pestaña ‘Reporte’ para elegir el tipo de reporte que queremos exportar.



Los formatos más útiles son xml y html, ambos son personalizables y luego fácilmente se los puede convertir en pdf, como veremos en la siguiente práctica.

Así es como se ve el reporte exportado en html:



ZAP Scanning Report

Summary of Alerts

Risk Level	Number of Alerts
High	4
Medium	3
Low	6
Informational	4

Alert Detail

High (Medium)	Insecure Component - Microsoft-IIS 7.5
Description	Based on passive analysis of the response, insecure component Microsoft-IIS 7.5 appears to be in use. The highest noted CVSS rating for this product version is 10. In total, 5 vulnerabilities were noted. Some Linux distributions such as Red Hat employ the practice of retaining old version numbers when security fixes are "backported". These cases are noted as "False Positives", but should be manually verified.
URL	https://hack.me/
Evidence	Microsoft-IIS/7.5

Si editamos el código fuente podría quedarnos mucho más presentable, ya que es html puro y podemos moldear la tipografía, tamaño y color de letras, ubicación del contenido y todo lo que deseemos.

En el ejemplo sólo le agregamos una imagen y centramos el *‘Scanning Report’*.

```
<html>
<head>
<META http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>SniferL4bs Scanning Report</title>
</head>
<body text="#000000">
<center>
<img src="http://1.bp.blogspot.com/-5aXflm-MhlQ/Uq0MUCjlXjI/AAAAAAAADok/SeRZwIJTBNk/s1600/Sniferlabs.png">
<p>
<p>
<h1>
<strong>Scanning Report</strong>
</h1></center>
</p>
</p>
<br /><br />
<p>
<strong>Summary of Alerts</strong>
</p>
<table width="45%" border="0">
<tr bgcolor="#666666">
<td width="45%" height="24"><strong><font color="#FFFFFF" size="2" face="Arial, Helvetica, sans-serif">Risk
      Level</font></strong></td><td width="55%" align="center"><strong><font color="#FFFFFF" size="2" face="Arial,
      of Alerts</font></strong></td>
</tr>
```

Lo convertimos a pdf, en algún sitio online si no tenemos ganas de programar un conversor y con muy poquito lo mejoramos bastante.

Scanning Report

Summary of Alerts

Risk Level	Number of Alerts
High	4
Medium	3
Low	6
Informational	4

Alert Detail

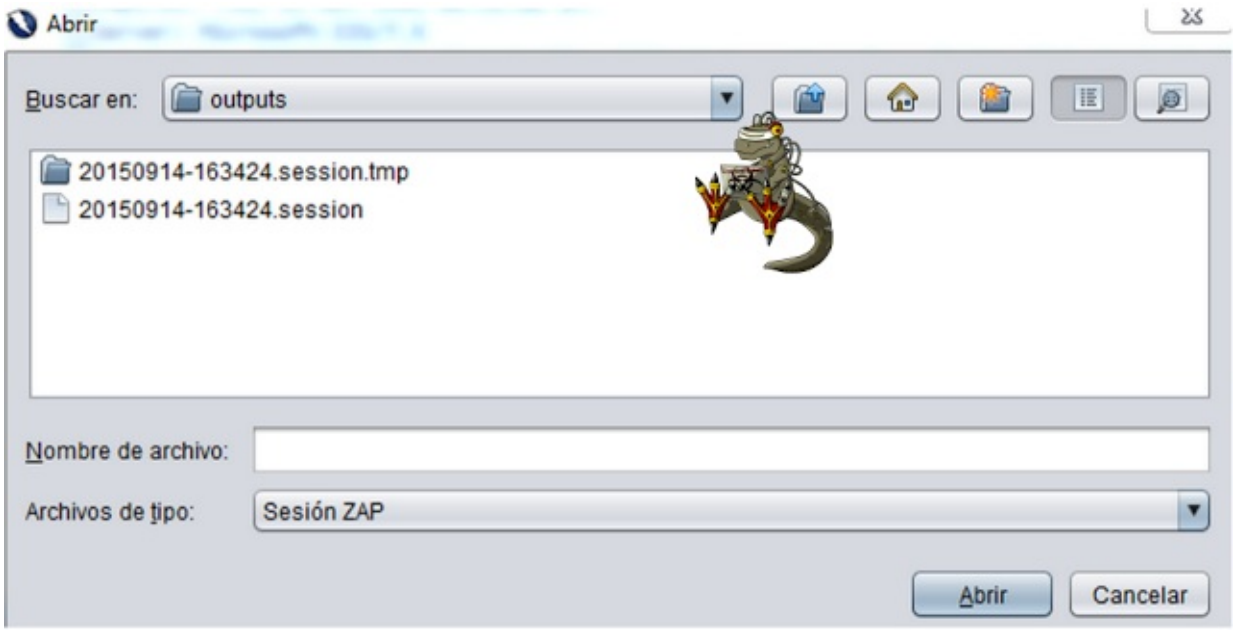
High (Medium)	Insecure Component - Microsoft-IIS 7.5
Description	Based on passive analysis of the response, insecure component Microsoft-IIS 7.5 appears to be in use. The highest noted CVSS rating for this product version is 10. In total, 5 vulnerabilities were noted. Some Linux distributions such as Red Hat employ the practice of retaining old version numbers when security fixes are "backported". These cases are noted as "False Positives", but should be manually verified.

Exactamente lo mismo se puede hacer con el formato xml.

Comparar reportes

En realidad, más que los reportes, lo que comparamos son las sesiones producto de nuestro análisis. Una vez que terminamos de realizar la auditoría, es recomendable guardar el reporte, en html por ejemplo, y la sesión sobre la que hemos trabajado. De ésta manera, si identificamos vulnerabilidades que deben ser remediadas, cuando auditamos nuevamente podemos corroborar con la sesión y con el reporte si lo los responsables han solucionado las vulnerabilidades reportadas o si siguen presentes a la espera de un atacante externo.

En la pestaña *'Reporte'* tenemos la opción de *'Comparar con otra sesión'*, se nos abre una ventana para poder elegir la sesión a comparar



Cierre

Con esto finalizamos este Ebook de Zap, espero que lo hayan disfrutado y les hayan sido útiles al menos para interiorizarse un poco con la herramienta, en lo posible trataremos de mejorar e ir actualizando tanto en contenido como algun error ortográfico o de contenido.

Por último, muchas gracias José por brindarme este espacio para compartir sobre ésta tool y felicitaciones por los 4 años de SniferL4bs!

Autor: Eric Balderrama

Twitter: @BalderramaEric

```
Ocupate viviendo o muriendo.
```