Monitorizacion web

Volver al Indice

Monitorización de experiencia de usuario Web

Introducción

En la versión Enterprise es posible monitorizar una Web utilizando el componente WEB Server, también llamado Goliat Server.

Esta funcionalidad proviene de un antiguo proyecto del fundador de Pandora FMS. Goliat F.I.S.T fue un proyecto opensource para realizar auditorías de carga dinámicas sobre servicios web. Todavía se puede encontrar el código fuente (del año 2002) y se dejó de mantener más o menos en el 2010 [1].

En Pandora FMS, funciona como un servidor independiente, similar al servidor de red, al servidor WMI o al servidor de plugins remotos. Este sistema opera bajo el principio de *transacción web*, donde cada transacción completa contra una (o varias páginas WEB) esta definida por uno o más pasos consecutivos, que deben concluir satisfactoriamente para considerar que la transacción ha terminado con éxito. La ejecución de una *transacción web* reproduce fielmente el proceso de navegación completo que puede incluir aspectos como autenticarse en un formulario, hacer clic en una opción del menú, rellenar un formulario, verificando que cada paso devuelve una cadena de texto concreta.

Cualquier fallo en un punto del proceso, daría como resultado un fallo en la comprobación. La transacción completa incluye la descarga de todos los recursos (gráficos, animaciones, etc.) que contempla la navegación real. Además de realizar comprobaciones de funcionamiento y de tiempo de respuesta, es posible extraer valores de las páginas web para luego procesarlos.

Goliat es capaz de monitorizar tanto HTTP como HTTPS de forma transparente para el usuario, soporta gestión de sesiones a través de cookies, paso de parámetros, y por supuesto, la descarga de los recursos asociados a cada página. También tiene **limitaciones importantes como son la gestión dinámica de javascript en tiempo de ejecución**. Para transacciones web más complejas, Pandora FMS dispone de otro componente mucho más potente (y complejo) llamado Monitorización WUX (**Web User Experience**), para más información siga el enlace: [2]

Instalación y configuración

Para poder utilizar Goliat, primero debe activarlo en el servidor enterprise de Pandora FMS:

webserver 1

En función del número de peticiones que quiera hacer puede que tenga que aumentar el número de hilos y el timeout por defecto:

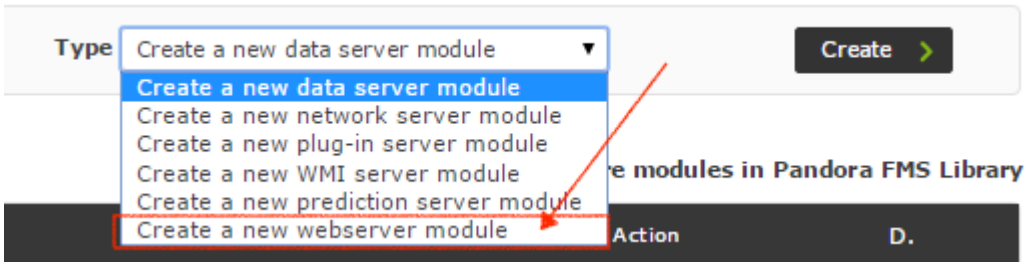
web_threads 1
web_timeout 60

Existe un token de configuración avanzado que le permitirá cambiar el tipo de librería que utiliza por debajo Goliat, LWP o CURL. Por defecto se utiliza LWP aunque si tiene algún problema (conexiones SSL, timeouts largos o problemas de concurrencia en el caso de muchos hilos), puede cambiar a CURL:

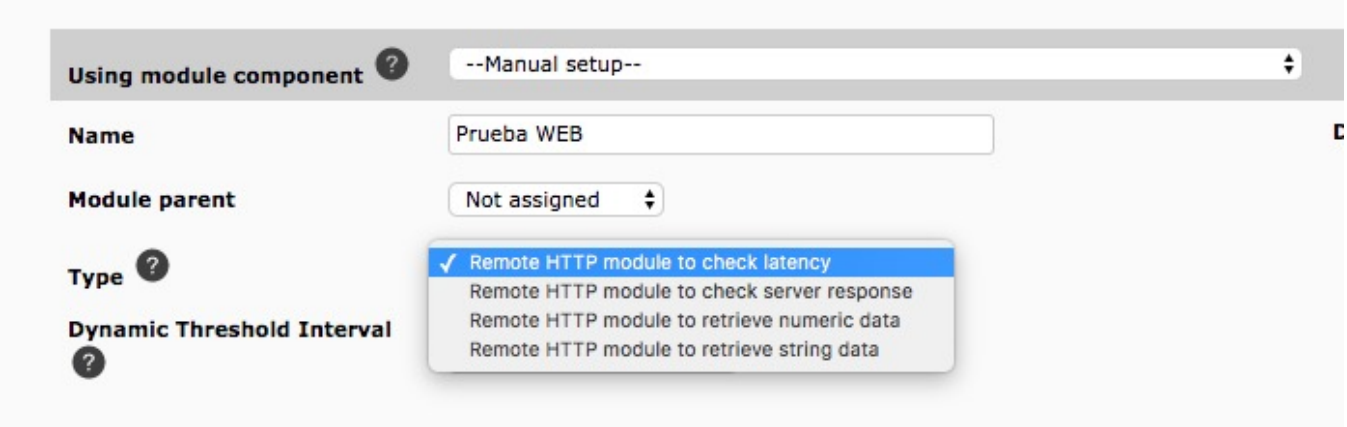
web_engine curl

Creación de módulos web

Para monitorizar de forma remota una página web, una vez creado el agente, pulse sobre la pestaña superior de los módulos (Modules). En ella, seleccione crear un nuevo módulo de servidor Web (webserver module) y pulse el botón *Create*:



Una vez que se pulse el botón *Create*, se mostrará un formulario en el que se han de rellenar los campos necesarios para poder monitorizar una web. El nombre y sobre todo, el tipo de chequeo WEB, son lo más básico:



Estos cuatro tipos sirven para diferentes cosas:

*__Remote HTTP module to check latency__: obtiene el tiempo total que transcurre desde la primera petición hasta que se comprueba la última (en una prueba WEB existe una o varias peticiones intermedias que completan la transacción). Si en la definición del chequeo se ha definido que la transacción se realice más de una vez, se utilizará la media del tiempo de cada petición.

*__Remote HTTP module to check server response__: obtiene un 1 (OK) o un 0 (CRITICAL) como resultado de comprobar toda la transacción. Si existen varios intentos, pero al menos uno de ellos tiene fallo, se considera que la prueba en su conjunto, también falla. Precisamente el número de intentos se utiliza en ocasiones para evitar falsos positivos. En el caso de que desee realizar la prueba varias veces por si una falla, utilice el campo *reintentos* (ver campos avanzados, más abajo).

*__Remote HTTP module to retrieve numeric data__: obtiene un valor numérico, *parseando* la respuesta HTTP utilizando una expresión regular para obtener ese valor.

*__Remote HTTP module to retrieve string data__: similar al anterior, pero con una cadena de texto.

Web checks

Este campo esencial, define la comprobación WEB que se va a realizar. Esta se define en uno o más pasos, o peticiones simples. Esas peticiones simples se deben escribir en un formato especial en el campo Web checks. Las comprobaciones se inician con la etiqueta *task_begin* y finalizan con la etiqueta *task_end*.

Un ejemplo completo de transacción sencillo seria el siguiente:

task_begin
get http://apache.org/
cookie 0
resource 0
check_string Apache Software Foundation
task_end

En este ejemplo básico, estamos comprobando si existe una cadena en una página web, para ello está la variable **check_string**. Esta variable no permite comprobar HTML en sí, solo busca subcadenas de texto. Estamos buscando "Apache Software Foundation" en la web http://apache.org. Si existe esa cadena de texto, el chequeo devolverá OK (si es de tipo *Remote HTTP module to check server response*)

Para asegurarse de que una cadena no existe en una página web, puede utilizar la variable 'check_not_string':

```
check_not_string Section 3
```

Para la comprobación de formularios existen varias variables adicionales:

*__resource (1 ó 0)__: descarga todos los recursos de la web (imágenes, vídeos, etc)
*__cookie (1 ó 0)__: mantiene una cookie, o una sesión abierta para comprobaciones posteriores
*__variable_name__ : nombre de una variable en un formulario
*__variable_value__: valor de la variable anterior en el formulario

Con estas variables se podrán enviar datos a formularios y comprobar que funcionan correctamente. En algunos casos de redirección de dominios los chequeos podrían no funcionar, para solucionarlo tendrá que crear el módulo apuntando al dominio final.

Los argumentos que toma la sintaxis de "check_string" no son cadenas de texto normales, son *expresiones regulares*. Es decir, si busca la cadena "Pandora FMS (4.0)" tendrá que buscarla con una expresión regular, p.e: Pandora FMS (4.0). Esto le permite hacer búsquedas mucho más potentes, pero debe tener en cuenta que cualquier carácter que no sea una letra o un número tendrá que ser escapado con \.

Comprobar tiempo de carga de una web

Si queremos comprobar el tiempo de respuesta o latencia de una página web sólo tenemos que seleccionar el tipo de módulo *Remote HTTP module to check latency*. Por ejemplo si queremos conocer la latencia de la carga de la página web http://pandorafms.com, el código sería tan simple como:

task_begin
get http://pandorafms.com
task_end

Podemos añadir los token de configuración **resource 1** para que el tiempo de descarga que calcule sea descargando todos los recursos (javascript, CSS, imágenes, etc), calculando así un dato mas real.

El tiempo de descarga de la web NO es el tiempo que tarda en verse una web en un navegador, ya que eso suele depender del tiempo de carga del Javascript, y Goliat, descarga el javascript, pero no lo ejecuta.

Chequeos a través de un Proxy

Los chequeos web también soportan el uso de proxy. Para configurarlo tiene que añadir la URL del proxy en la casilla que está encontrará al pulsar en *Advanced options*:

Por ejemplo la URL podría ser:

http://proxy.domain.com:8080

Si el proxy requiere autenticación la URL sería como la siguiente:

http://my_user:my_pwd@proxy.domain.com:8080

Obteniendo datos de una página web

Quizás no queremos saber si una Web específica está funcionando o cuanto tarda, si no que queremos obtener un valor en tiempo real, como por ejemplo, el valor en bolsa de Google. Para esto, emplearemos un módulo *Remote HTTP module to retrieve numeric data* con la expresión regular adecuada:

```
task_begin
get http://finance.google.com/finance/info?client=ig&q=NASDAQ%3aGOOG
get_content \d+\.\d+
task_end
```

La salida será algo parecido a esto:

También es posible especificar una expresión regular más compleja para recoger datos de respuestas HTTP más complejos con el token de configuracion *get_content_advanced*:

```
task_begin
get http://finance.yahoo.com/q?s=GOOG
get_content_advanced <span id="yfs_l84_goog">([\d\.]+)</span>
task_end
```

La parte de la expresión regular (definida en *get_content_advanced*) a devolver debe estar encerrada entre paréntesis.

Para configurar los umbrales que nos dispararán los estados de advertencia o crítico, utilizaremos la configuración del módulo para comprobar que la cadena recibida coincide con lo esperado.

Comprobación de formulario en una página web

Una comprobación más interesante es la comprobación de un formulario web. No obstante es mucho más compleja que la simple comprobación de un texto en una página web. Esta comprobación de ejemplo usará la propia consola de Pandora, iniciará sesión, y comprobará que efectivamente ha sido capaz de ello, verificando un texto en la sección de workspace donde muestra los datos del propio usuario que ha iniciado sesión. Si se trata de una consola por defecto, el usuario admin contiene la descripción "Admin Pandora".

Para poder realizar este tipo de comprobaciones se ha de tener las credenciales necesarias para poder iniciar sesión (ya que usamos esos valores para "enviarlos" al formulario HTML. Además, se deberá ir a la página y obtener el código HTML para poder ver los nombres de las variables, luego es preciso tener conocimientos mínimos de HTML para entender como funciona Goliat. En este ejemplo estamos usando "admin" con la password "pandora" que son las credenciales por defecto. Deberíamos haberlas cambiado, si no es así, ¡este es un momento bueno para hacerlo!

Lo ideal a la hora de diseñar una prueba transaccional WEB con varios pasos es ir probándola paso a paso, por si en uno de los pasos se nos ha escapado algo.

Supongamos que la URL de login de nuestra Consola de Pandora está en:

http://192.168.70.116/pandora_console/

Analizando su código HTML, se observa que las variables del formulario de login son:

- *nick*: nombre del usuario
- *pass*: contraseña para el usuario

Se deberán usar las variables variable_name y variable_value conjuntas para poder validar el formulario.

El primer paso es acceder al formulario, enviar el usuario y el password y autenticarse (determinar el éxito de esa autenticación lo veremos en el segundo paso).

```
task_begin
post http://192.168.70.116/pandora_console/index.php?login=1
variable_name nick
variable_value admin
variable_name pass
variable_value pandora
cookie 1
resource 1
task_end
```

Con la tarea anterior se habría logrado acceder a la página web y validarse en ella, ahora se comprobará que efectivamente se está registrado en la página buscando algo en ella que sólo se puede ver estando registrado. Utilizamos el token "cookie 1" para mantener la persistencia de las cookies obtenidas en el paso anterior. Sin ellas, no podremos "simular" una sesión.

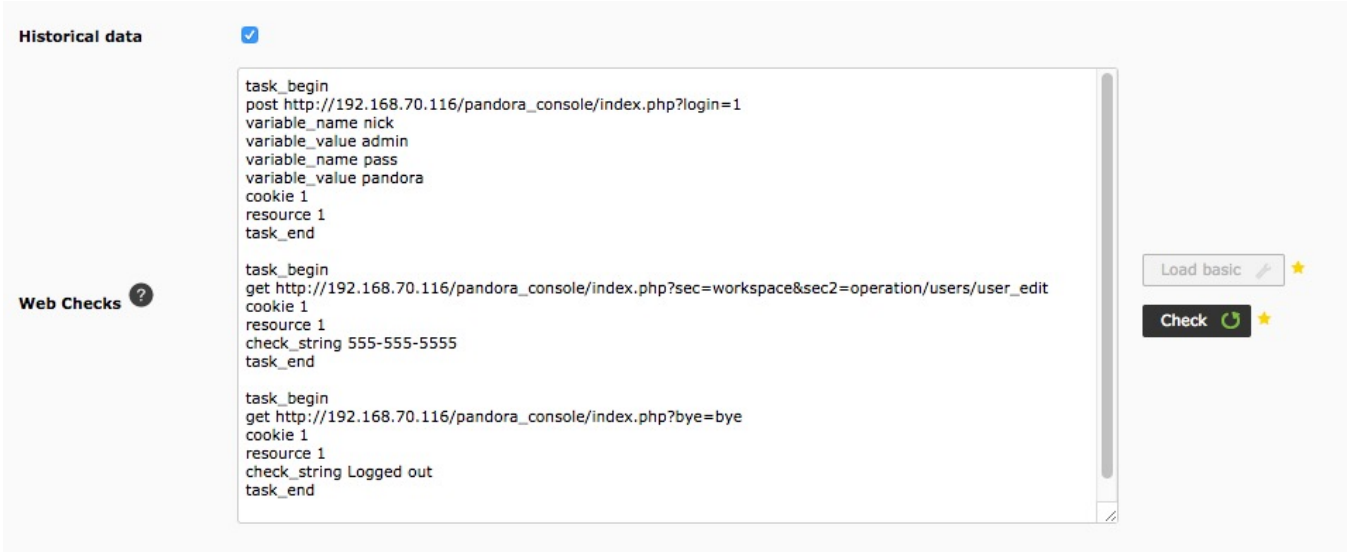
En el segundo paso accederemos a la página de detalles del usuario y buscaremos el teléfono, que por defecto para el usuario "admin" es 555-555-555. Si lo podemos ver es que hemos iniciado sesión correctamente en la consola:

```
task_begin
get http://192.168.70.116/pandora_console/index.php?sec=workspace&sec2=operation/users/user_edit
cookie 1
resource 1
check_string 555-555-5555
task_end
```

Y para terminar, desconectaremos de la consola y buscaremos el mensaje de "desconexión (log out)":

```
task_begin
get http://192.168.70.116/pandora_console/index.php?bye=bye
cookie 1
resource 1
check_string Logged out
task_end
```

Con lo cual la comprobación total quedaría en Pandora FMS como sigue:



Comportamiento de las peticiones WEB

Los campos de las propiedades avanzadas son similares a los de otros tipos de módulos, aunque existen algunos campos diferentes y propios de los chequeos WEB:

Timeout

Es el tiempo de expiración durante la petición, si se supera éste tiempo la petición de comprobación se descartará.

Agent browser id

Es el identificador de navegador web que usar, ya que determinadas páginas sólo aceptan algunos navegadores web (consultar zytrax.com para obtener más información).

Requests

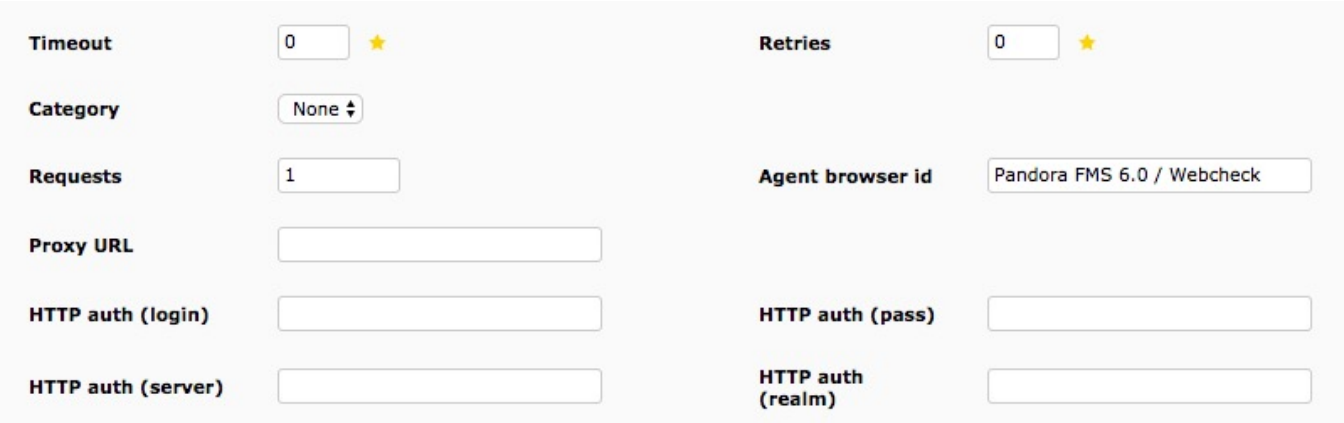
Pandora repetirá la comprobación el número de veces que se indique en este parámetro. Si una de las comprobaciones falla, la comprobación se dará como errónea. Dependiendo de la cantidad de comprobaciones en el módulo, se obtendrá un número determinado de páginas; es decir, si el módulo consta de tres comprobaciones, se descargarán tres páginas, y si en el campo Requests se ha establecido algún valor, entonces el número de descargas se multiplicará por éste. Es importante tenerlo en cuenta para saber el tiempo total que tardará el módulo en completar las operaciones.

Retries

No tiene que ver con "requests" e implica lo contrario, es decir, que si su valor es > 1, y falla la primera vez, reintentará un número determinado de veces hasta que lo consiga. Por ejemplo con retries = 2 y Requests = 1, si la primera prueba fallara, lo reintentaría una vez más, y si a la segunda funciona, el chequeo se daría como válido. Si fuera Requests = 2, y Retries = 1, realizaría dos chequeos, pero si cualquiera de los dos fallara, daría la prueba como fallida.

Utilizando autenticación HTTP Simple

Algunas páginas pueden requerir autenticación simple HTTP. Eso no tiene nada que ver con un usuario y password en un formulario. La autenticación HTTP es aquella en la que "salta" una ventana del navegador, informando que el sitio "xxxx" nos pide unas credenciales.



Se puede configurar en las opciones avanzadas del chequeo (según se puede ver en la captura anterior) o directamente en la definición de la tarea WEB con los siguientes token de configuración:

- http_auth_serverport - Dominio y el puerto web
- http_auth_realm - Nombre de la zona (realm) de autenticación.
- http_auth_user - Usuario
- http_auth_pass - Password

Un ejemplo completo seria el siguiente:

task_begin
get http://artica.es/pandoraupdate4/ui/
cookie 1
resource 1
check_string Pandora FMS Update Manager (4.0)
http_auth_serverport artica.es:80
http_auth_realm Private area
http_auth_user admin
http_auth_pass xxxx
task_end

Monitorizacion de webservices

Con Pandora FMS y Goliat se pueden monitorizar API's REST, no así API's mas complejas basadas en protocolos como SOAP o XMLRPC.

Por ejemplo, supongamos que quiero comprobar una API con esta llamada, que devuelve un numero entero (de 0 a infinito) en el caso de que funcione. NO devolver nada seria un error:

task_begin
get http://artica.es/integria/include/api.php?user=slerena&pass=xxxx&op=get_stats&params=opened,,1
check_string \n0-9+
task_end

Esto me devuelve una respueta del tipo:

```
HTTP/1.1 200 OK
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Connection: close
Date: Mon, 13 May 2013 15:39:27 GMT
Pragma: no-cache
Server: Apache
Vary: Accept-Encoding
Content-Type: text/html
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Client-Date: Mon, 13 May 2013 15:39:27 GMT
Client-Peer: 64.90.57.215:80
Client-Response-Num: 1
Client-Transfer-Encoding: chunked
Set-Cookie: a81d4c5e530ad73e256b7729246d3d2c=pcasWqI6pZzT2x2AuWo602; path=/

0
```

Mediante la comprobacion de la salida con una expresion regular, puede verificar que esta todo ok. Para respuestas mas complejas, deberia usar otras expresiones regulares.

Monitorización https

Goliat puede comprobar tanto http como https. Para poder hacer comprobaciones sobre web seguro (https) basta con especificar este protocolo en la URL, por ejemplo:

```
task_begin
get https://www.google.com/accounts/ServiceLogin?service=mail&passive=true&rm=false&continue=https%3A%2F%2Fmail.
google.com%2Fmail%2F%3Fui%3Dhtml%26zy%3Dl&bsv=zpwhtygjntrz&ss=1&scc=1&ltmpl=default&ltmplcache=2
cookie 1
resource 0
check_string Google
task_end
```

Soporte IPv6

Goliat soporta IPv6, aunque necesita usar direcciones FQDN, esto significa que las 'URL's deben tener nombres completos (p.e: ipv6.google.com). La representación numérica de direcciones IPv6 (pe: :1, 6800:4004:803::1014 etc..) no es válida para realizar chequeos IPv6 con Goliat.

Opciones avanzadas

Modificando cabeceras HTTP

Con la opción *header* se pueden modificar campos de la cabecera HTTP o crear campos personalizados. Por ejemplo, para cambiar el campo *Host* de la cabecera HTTP:

task_begin
get http://192.168.1.5/index.php
header Host 192.168.1.1
task_end

Depurando chequeos web

Se pueden depurar los chequeos web añadiendo la opción *debug <log_file>*. Se crearán dos ficheros log_file.req y log_file.res con los contenidos de la petición HTTP y la respuesta respectivamente. Por ejemplo:

task_begin
get http://192.168.1.5/index.php
debug /tmp/request.log
task_end

El anterior chequeo web creará los ficheros */tmp/request.log.req* y */tmp/request.log.res*.

Utilizando Curl en vez de LWP

La librería LWP puede dar problemas cuando muchos hilos llevan a cabo peticiones HTTPS (debido a una limitación de OpenSSL). Para solucionar este problema, edite el fichero */etc/pandora/pandora_server.conf* y añada la siguiente línea:

web_engine curl

Reinicie el servidor de Pandora FMS, y el binario de Curl se utilizará llevar a cabo las comprobaciones web en vez de LWP.

Monitorización transaccional distribuida con Selenium

Además de la funcionalidad que ofrece Goliat, integrado en Pandora FMS, existe (*En la version enterprise*) otra forma de realizar una monitorización transaccional con Pandora FMS, pero desplegada a modo de "agente" en sistemas diferentes al servidor, incluso en redes no accessibles.

Se utiliza Selenium [3] como motor ~~en vez de Goliat~~, y este soporta una navegación "por clics", mucho mas fina que la que ofrece Goliat. El plugin de agente de Selenium interactúa con el servidor de Selenium. y este a su vez con el navegador del sistema. Esto hace que se puedan realizar navegaciones usando Chrome, Firefox o IE indistintamente. Funciona en Windows o en Linux, y permite realizar cálculos de latencia, comprobacion de cadenas y **soporta la navegación con javascript pesado, applets java, flash o cualquier otra tecnología soportada en el navegador**.

La documentación de Selenium, por ser extensa y específica de dicha tecnología, está disponible en el plugin enterprise de Selenium, que puede encontrar en la libreria de modulos de pandorafms.com.

Volver al Indice