Monitorización Web

Monitorización Web

Monitorización Web clásica

Introducción

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 está definida por uno o más pasos consecutivos, que deben concluir satisfactoriamente para considerar que la transacción ha terminado con éxito.

- Web server 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).

Instalación y configuración

Se debe activar el Web server y reiniciar el Pandora FMS server. Para activar el Web server se debe modificar en el fichero de configuración del PFMS server:

```
webserver 1
```

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

```
web_threads 1
web_timeout 60

# Use curl or LWP
web_engine curl
```

Pandora FMS cuenta con protección contra CSRF y puede ocurrir que los chequeos web, al ser depurados, obtenga este mensaje:
`Cannot verify the origin of the request`
Tenga en cuenta esta protección para considerar el uso de "Monitorización WUX".

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 Create a new webserver module y pulse el botón Create:

Se debe seleccionar el tipo de chequeo WEB:

- 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 existen 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 falla, se considera que la prueba en su conjunto, también falla. Precisamente, el número de intentos se utiliza en ocasiones para evitar falsos positivos, para ello utilice el campo *reintentos* en campos avanzados.
- 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: Análogo al punto anterior pero con una cadena de texto.
- Remote HTTP module to check server status code: Por medio de la herramienta curl debidamente habilitada con el *token* de configuración `web_engine curl` se puede retribuir las cabeceras HTTP.

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.. Las comprobaciones se inician con la etiqueta `task_begin` y finalizan con la etiqueta `task_end`.

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.

En algunos casos de re-dirección de dominios los chequeos podrían no funcionar. Una manera de solucionarlo es modificar el módulo apuntando al dominio final, por ejemplo curl -L

Comprobar tiempo de carga de una web

Para comprobar el tiempo de respuesta o latencia de una página web se selecciona el tipo de módulo Remote HTTP module to check latency. Ejemplo:

```
task_begin
get https://pandorafms.com
```

```
task_end
```

- Para que el tiempo de descarga incluya todos los recursos (JavaScript, CSS, imágenes, etcétera) se debe añadir `resource 1` en una línea antes de `task_end`.
- Los chequeos web también soportan el uso de *proxy* en el *token* Proxy URL.

El tiempo de descarga de la web no es el tiempo que tarda en visualizarse una web en un navegador, ya que esto suele depender del tiempo de carga del JavaScript.

Comprobación de formulario en una página web

Una comprobación de formulario es mucho más compleja que la simple comprobación de un texto en una página web. Para poder realizar este tipo de comprobaciones se deben tener las credenciales necesarias. Además, es necesario *ir* a la página y obtener el código HTML para obtener los nombres de las variables, y luego es preciso tener conocimientos mínimos de HTML para introducir la consulta para el web server.

El método práctico para diseñar una prueba transaccional WEB con varios pasos es probarlos uno por uno en modo de depuración de errores.

Utilizando autenticación HTTP Simple

Algunas páginas pueden requerir autenticación simple HTTP. Generalmente es utilizada como una comprobación rápida, un *saludo* de seguridad mínima que permite acceder a comprobaciones de seguridad más avanzadas (cifrado, persistencia de datos, etcétera).

- El uso de comillas en la contraseña para `http_auth_pass` no está soportado.
- Evite el uso de comillas simples.

Monitorización de web services y API

Se pueden monitorizar API de tipo REST, exceptuando los tipos de API más complejas basadas en protocolos como SOAP o XMLRPC.

Mediante la comprobación de la salida con una expresión regular, puede verificar que esté todo correcto. Ejemplo:

```
task_begin
get https://swapi.dev/api/planets/1/
get_content Tatooine
```

```
task_end
```

Para respuestas más complejas se debe usar otras expresiones regulares y el *token* `get_content_advanced`.

- Es importante definir correctamente los grupos de captura entre paréntesis para que la llamada se realice de manera adecuada.
- Al hacer llamadas a la API es importante tener en cuenta que la API destino debe tener permisos para poder ser consultada.

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
```

Utilizando Curl en vez de LWP

La utilidad LWP puede dar problemas cuando muchos hilos llevan a cabo peticiones HTTPS (debido a una limitación de OpenSSL). La alternativa es utilizar la herramienta curl. 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 para llevar a cabo las comprobaciones web en vez de LWP.

Monitorización transaccional avanzada

Además de la funcionalidad que ofrece Web server PFMS, existen otras maneras de realizar una monitorización transaccional web.

- De manera distribuida (UX), desplegada en modo de “agente” en sistemas diferentes al servidor, incluso en redes no accesibles.
- De manera centralizada (WUX).

Volver al índice de documentación de Pandora FMS