

Proyecto Final: Herramienta para monitoreo de bitácoras relacionadas con servicios web

Integrantes: José Juan Armenta Segura Diego Alfonso Serrano Guillén Responsable del Proyecto: Angie Aguilar Domínguez



02 de NOVIEMBRE de 2016

UNAM-CERT

Contenido

[**1.** **Introducción** 1](#_Toc466375099)

[**2.** **Metodología de recolección de información** 3](#_Toc466375100)

[**3.** **Análisis de bitácoras de Apache y PostgreSQL** 5](#_Toc466375102)

[**3.1.** **SQLi** 10](#_Toc466375103)

[**3.2.** **Cross Site Scripting Reflejado** 13](#_Toc466375104)

[**3.3.** **Web Crawling / Spidering** 16](#_Toc466375105)

[**3.4.** **Path Traversal** 20](#_Toc466375106)

[**3.5.** **Defacement** 23](#_Toc466375107)

[**4.** **Análisis de bitácoras de ModSecurity** 26](#_Toc466375108)

[**5.** **Reporte de hallazgos** 27](#_Toc466375109)

[**6.** **Consideraciones previas y requisitos** 31](#_Toc466375110)

[**7.** **Instalación** 33](#_Toc466375111)

[**8.** **Configuración** 39](#_Toc466375112)

[**8.1.** **Archivo de configuración** 39](#_Toc466375121)

[**8.2** **Dar de alta los sitios Web instalados** 41](#_Toc466375128)

[**9.** **Modo de uso** 42](#_Toc466375129)

[**10.** **Mejoras a futuro** 44](#_Toc466375130)

[**11.** **Anexo A** 45](#_Toc466375131)

[**11.1.** **Lista negra para detección de herramientas** 45](#_Toc466375132)

[**11.2.** **Lista negra para SQL injection** 46](#_Toc466375133)

[**11.4.** **Lista negra para Bots** 48](#_Toc466375134)

[**11.5.** **Lista negra para Path Traversal con variables en la URL** 49](#_Toc466375135)

[**11.6.** **Lista negra para Path Traversal sin variables en la URL** 50](#_Toc466375136)

[**12.** **Referencias** 51](#_Toc466375137)

# **Introducción**

Esta herramienta fue desarrollada de forma modular, en donde algunos módulos fueron escritos en Perl y en conjunto son controlados por otro componente escrito en Python 2.7, el objetivo de la aplicación es analizar la información de las bitácoras de un servidor Apache 2.2, otro servidor de Base de datos PostgreSQL 9.1 y por último un WAF (ModSecurity 2.9), a partir de las bitácoras la herramienta debe ser capaz de detectar y clasificar algún ataque a los virtual hosts del servidor web. Además si el servidor tiene instalada más de un sitio, la aplicación es capaz de analizar todos los sitios que se tienen siempre y cuando estén definidos en un archivo de configuración. Para este proyecto, se realizó una infraestructura que cuenta con tres servidores con un sistema operativo Debian 7, en el servidor Web se tiene instalado un gestor de contenidos Drupal la cual se apoya del servidor de base de datos:



**Servidor Web, con Apache y Drupal**



**Servidor de Base de Datos PostgreSQL**

**Servidor WAF**

Los ataques que se detectarán son:

* SQL Injection.
* Cross Site Scripting reflejado.
* Path Traversal.
* Crawling.
* Defacement.

Además si el servidor WAF está en modo detección *“DetectionOnly”* la herramienta detectará las alertas que ModSecurity genere. Una vez que la aplicación detecte algún evento mencionado anteriormente, se enviará un correo electrónico informando acerca del problema, incluyendo algunos datos como la marca de tiempo, dirección IP, método HTTP, payload, referer, código de estado HTTP, tamaño en bytes y el User-Agent.

La aplicación cuenta con algunos archivos de configuración los cuales contienen direcciones IP, nombres de usuario, rutas e información que determine el comportamiento del análisis de los distintos ataques mencionados anteriormente.

Para la recolección de los eventos registrados en las bitácoras de cada uno de los servidores, la herramienta se apoya del protocolo SSH, se realiza una conexión en el servidor Web y el de Base de datos y verificará las bitácoras de manera periódica, las nuevas líneas que se han agregado desde el inicio del monitoreo hasta haber transcurrido cierta cantidad de tiempo, la cual también se tiene establecido en uno de los archivos de configuración. Una vez que se ejecuta el comando para obtener las nuevas líneas el resultado que se obtenga redirige a un archivo y es transferido y almacenado en el servidor WAF. Lo anterior se resume en el siguiente diagrama:

# **Metodología de recolección de información**

Ésta parte se desarrolló en un script Python llamado *Proyecto.py*, el cual obtiene las bitácoras de cada servidor, se realiza una conexión desde el servidor WAF hacia los servidores Web y de Base de datos por medio de SSH, para ello se necesita un par llaves que creadas al momento de instalar la aplicación con el script installSSH.sh, las llaves son generadas a partir de un cifrado RSA. Una vez que se establece una conexión se crea un hilo para realizar lo siguiente de forma independiente:

1. Se consulta en cada servidor el número de líneas que tiene la bitácora.

Hilo 1

**SSH**



**SSH**



**SSH**

1. Transcurren 30 segundos para esperar que existan nuevas líneas en las bitácoras.
2. Después de haber transcurrido ese lapso de tiempo se vuelve a consultar el número de líneas que tiene la bitácora anterior.
3. Si hay un aumento en la cantidad de líneas que se realizó en el paso 1 con el paso 3, se extraen las líneas que esté dentro del rango, ejemplo: Si en el paso 1 habían 1000 líneas y en el paso 3 fueron 1,500 líneas se obtiene la bitácora a partir de la línea número 1000 hasta la 1,500
4. La salida es redirigida a un archivo de texto plano que se aloja en el servidor WAF.
5. Finalmente se cambia el formato de todos las bitácoras, por ejemplo:

Formato original del archivo access.log:

192.168.35.129 - - [25/Oct/2016:03:54:11 -0500] "GET /ejemplo.php?name=diego%3Cscript%3Ealert(%22XSS%22)%3C/script%3E HTTP/1.1" 200 332 "-" "Mozilla/5.0 (Windows NT 6.3; WOW64; rv:49.0) Gecko/20100101 Firefox/49.0"

Formato modificado del archivo access.log:

192.168.35.129<-->25/Oct/2016:03:54:11<-->GET<-->/ejemplo.php?name=diego%3Cscript%3Ealert(%22XSS%22)%3C/script%3E<-->HTTP/1.1<-->200<-->332<-->-<-->Mozilla/5.0 (Windows NT 6.3; WOW64; rv:49.0) Gecko/20100101 Firefox/49.0

Como se aprecia en el formato modificado, se agrega el separador <--> debido a que es más fácil de analizar el archivo, en la sección de análisis se detalla la lectura de este archivo con el formato modificado.



# **Análisis de bitácoras de Apache y PostgreSQL**

Una vez que se obtienen las bitácoras desde el script de Python (*Proyecto.py*), manda a llamar dos módulos desarrollados en Perl, el primer módulo recibe el nombre de *analizador.pl*, el cual siempre va a ejecutarse y su objetivo es detectar algún ataque por medio de las bitácoras generados por el servidor Web y el de Bases de datos, se desarrollaron unos submódulos para detectar los siguientes ataques:

* SQL Injection.
* XSS reflejado.
* Path Traversal.
* Crawling.
* Defacement.

Además realizará una correlación entre el archivo access.log y log de errores (servidor web o base de datos) continuación se presenta un diagrama de flujo del primer módulo:

Abrir el archivo access.log del servidor Apache

Leer línea del access.log

1

arreglo = Split (LineaLog, “<-->”)

¿Hay XSS?

¿Hay Path Traversal?

Llamar submódulo Defacement

¿Hay SQL Injection?

Llamar submódulo SQL Injection

1

Llamar submódulo XSS

Llamar submódulo Path Traversal

**Si**

**Si**

**Si**

Decodificar recurso

**No**

**No**

**No**

**Si**

**No**

**Si**

**No**

¿Hay Defacement?

¿Recurso codificado en base64 o en hexadecimal?

2

**Si**

Mandar correo electrónico.

Reportar evento(s).

**Si**

¿Hay reportes?

Detectar herramienta

Reportar evento.

**No**

**No**

**Si**

¿Hay Crawler?

Llamar submódulo Crawler

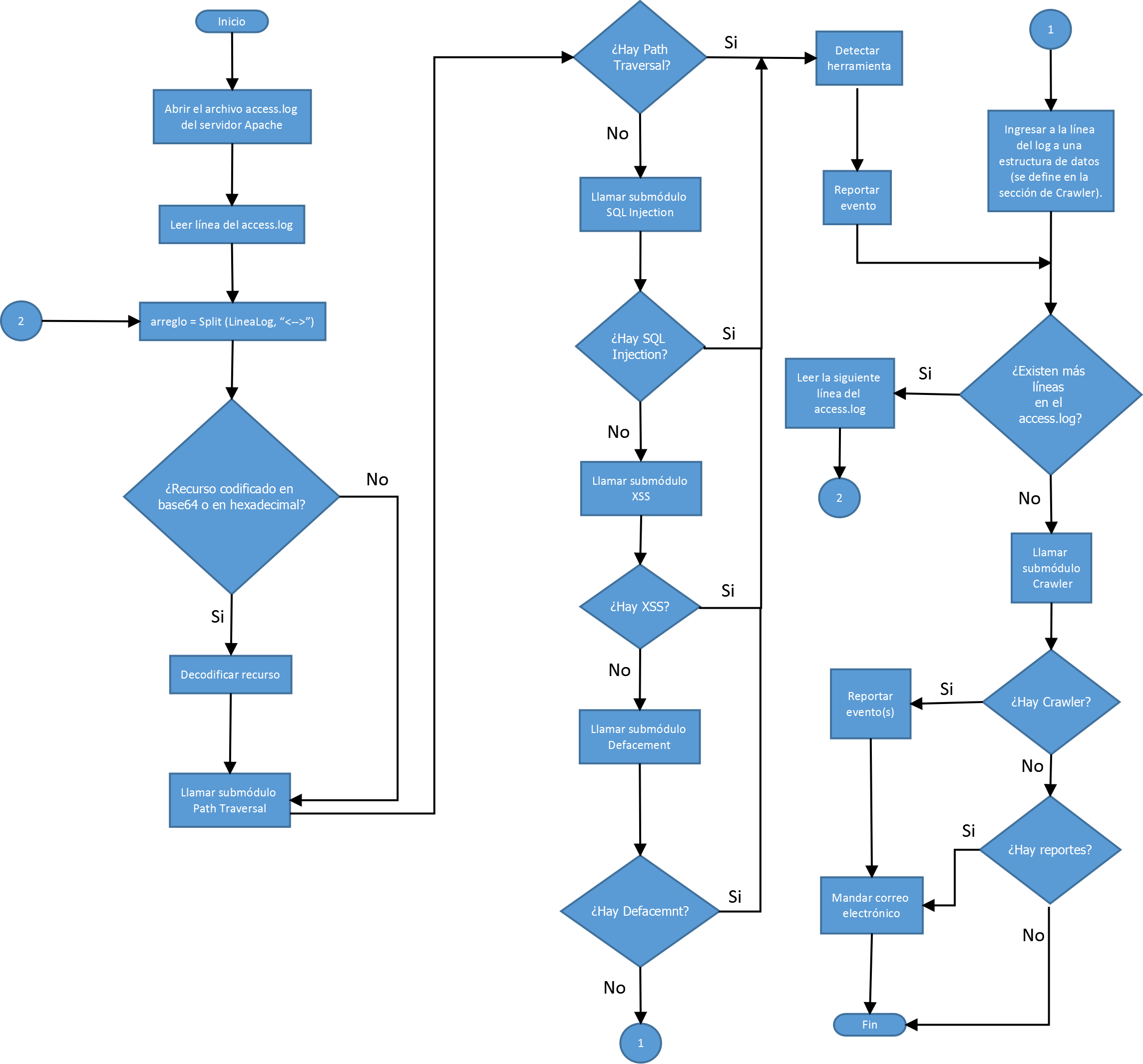
1

Leer la siguiente línea del access.log

¿Existen más líneas en el access.log?

Ingresar a la línea del log a una estructura de datos (se define en la sección de Crawler).

2



­

Esta herramienta tiene un archivo de configuración llamado config.conf el cual ayuda al usuario a personalizar el comportamiento de la herramienta, dicho archivo se modifica con cualquier editor de texto (vi, nano, etc.).

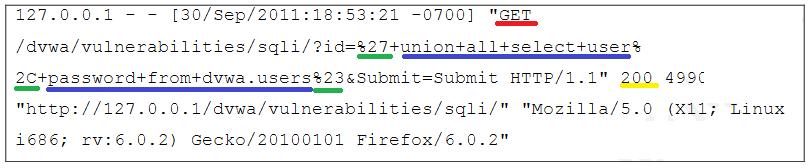
El diagrama de flujo puede sufrir cambios ya que en el archivo de configuración el usuario puede elegir los ataques que se desean detectar y los que no se detectarán, para tal efecto se tiene las siguientes variables las cuales si valen 1 se hará el análisis o de lo contrario si tienen un valor diferente a 1 no se realizarán:

* analizarXSS.
* analizarSQLi.
* analizarCrawler.
* analizarDefacement.
* analizarPatTrasversal.

Por otra parte, la detección de herramientas se realiza usando una lista negra llamada herramientas.txt, en donde se analiza la existencia de alguna de las palabras en la línea del log debido a que algunas herramientas usan su nombre en distintas partes de la petición por ejemplo: método HTTP, referer, directorios en la URL, User-Agent, etc. La lista negra se encuentra en el Anexo A. Además las palabras de lista se compara de manera case insensitive, por lo tanto no importará si existen palabras en mayúsculas o mínusculas.

## **SQLi**

Apoyándonos de la bitácora acces.log de Apache, se verificarán los valores de las variables que estén en el método GET de HTTP, para ello se va a extraer cada valor de las variables que existan y para ser analizados posteriormente; para verificar que sean valores sospechosos se debe de considerar que contengan sentencias SQL, valores que pueden estar codificados con el prefijo de porciento o puede tener letras mayúsculas y minúsculas. A continuación se presenta una traza de la bitácora de Apache:



Como vemos se tiene el método GET subrayado en color rojo, en verde aquellos caracteres codificados que intenta ofuscar o evadir alguna medida de seguridad, en color azul tenemos algunas instrucciones para el manejador de base de datos que modifica el query que la aplicación web hace, finalmente en color amarillo tenemos el código de respuesta. La detección será por medio de la combinación de expresiones regulares y una lista negra la cual se puede consultar en el anexo A sección 10.2.

Una vez que se detecte alguna anomalía, la herramienta hace una consulta en las bitácoras de PostgreSQL con el fin de validar que la sentencia o query que se haya ejecutado, para ello se toman los siguientes datos de las bitácoras de Apache para relacionarlos con la información de las bitácoras de PostgreSQL :

* Fecha y hora.
* Dirección IP.
* Fragmento del query o sentencia usada en el SQLi.

En el siguiente diagrama se muestra el flujo de detección de eventos para un ataque SQL injection, sin la capacidad de determinar el tipo de ataque efectuado sobre el sitio, el algoritmo usa como base la detección de sentencias SQL en la petición haciendo uso de los campos: recurso, referer y User-Agent , la detección de un evento está orientado al comportamiento y no a identificar el tipo de ataque SQL injection que se esté efectuando, este diagrama ejemplifica el proceso que sigue cada línea que se analiza la bitácora de Apache:

¿Existe alguna consulta SQL en el campo del recurso, referer o User-Agent?

**No**

**Hubo intento de SQL injection.**

**No se puede determinar intento de SQL injection**

**Si**

**No**

¿Hubo más de 30 (valor umbral definido por el usuario) errores en 30 segundos?

**Si**

**No**

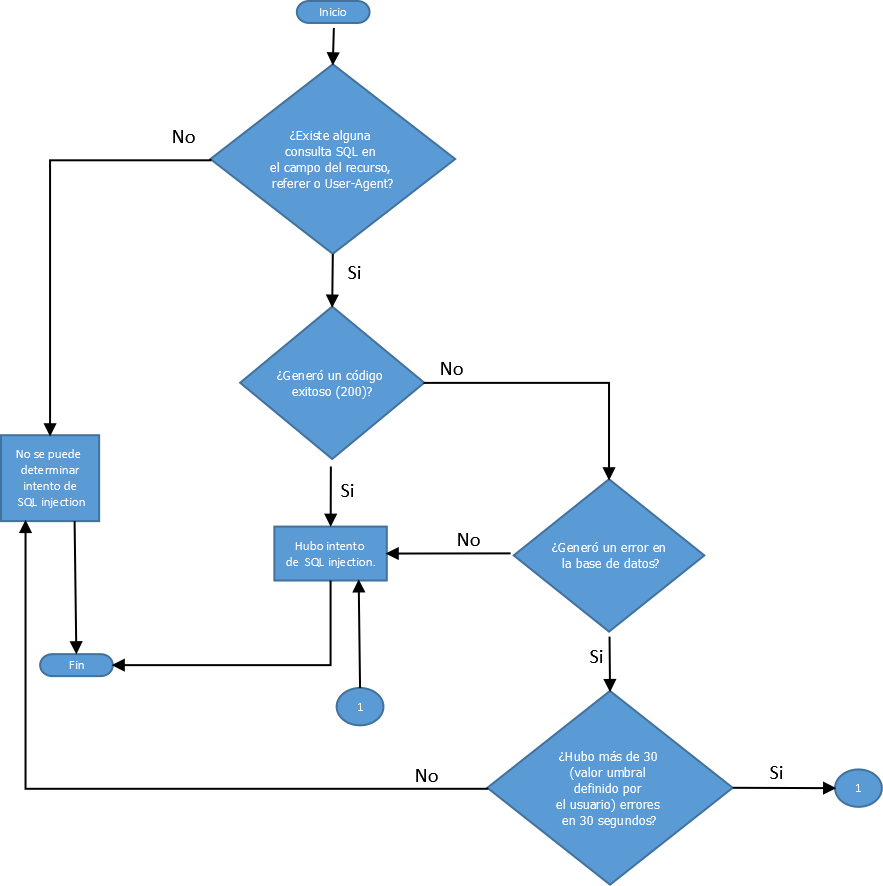
¿Generó un error en la base de datos?

**No**

**Si**

¿Generó un código exitoso (200)?

**Si**



|  |  |
| --- | --- |
| Elementos para la detección para ataques de SQL injection | |
| Elemento | **Descripción** |
| Recurso de la petición | Se debe encontrar indicios en una lista negra, se busca patrones de sentencias SQL en estos campos, en caso de ser positivos buscará los errores en la base de datos |
| User-agent de la petición |
| Referer de la petición |
| Código de respuesta de la petición | Si genera un código de respuesta positivo se toma como un intento de SQL injection, por lo tanto envía correo, en caso contrario revisa si se generó un error en la base de datos |
| Errores en la base de datos | Si la petición que se está analizando genero un error en la base de datos, se considera en un acumulador que cuenta durante un tiempo definido de 30 segundos buscando más peticiones similares, en dado caso de sobrepasar un valor de umbral (30 por defecto) se toma como un evento, esto se considera así porque muchas herramientas generan consultas para obtener información de la base de datos que se está intentando atacar.  En dado caso que no se encuentre la petición en la base de datos, se considera como evento igualmente. |

## **Cross Site Scripting Reflejado**

Este submódulo recibe varios argumentos, los más destacables son: la marca de tiempo, el recurso solicitado y las rutas de las listas negras; para el análisis este submódulo se enfocará en el recurso solicitado, solo se evaluarán aquellos ataques que se realicen por medio de las variables enviadas a través de la URL ya que es la única información que se encuentra en las bitácoras para detectar este tipo de ataque, y además se analizará si hay inyección de código HTML. A continuación se muestra el diagrama de flujo que se usa para el análisis:

¿Recurso solicitado tiene variables? 

**No**

No hubo un XSS.

**Si**

¿Existe una etiqueta HTML en alguna variable?

**Si**

**No**

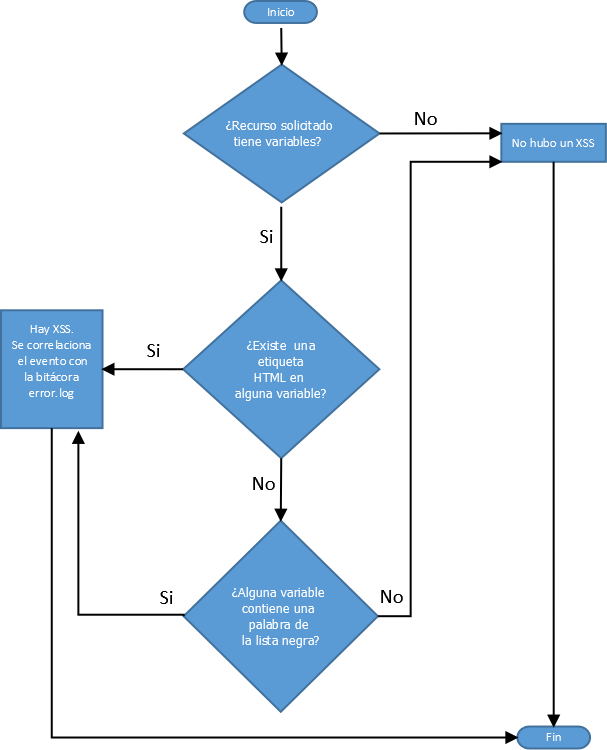
Hay XSS.

Se correlaciona el evento con la bitácora error.log

¿Alguna variable contiene una palabra de la lista negra?

**No**

**Si**



Se detectará un evento de tipo Cross Site Scripting Reflejado cuando se presenten las siguientes características:

|  |  |
| --- | --- |
| Elementos para la detección para ataques de Cross Site Scripting Reflejado | |
| Elementos | **Descripción** |
| Recurso solicitado con variables | Para ahorrar tiempo de ejecución se verifica la existencia de alguna variable en el recurso solicitado. |
| En las variables contiene alguna palabra de la lista negra. | Por defecto se tiene una lista negra la cual se encuentra en el Anexo A, donde el usuario puede agregar más palabras. Las palabras se analizan de forma case insensitive por lo que no se tomará en cuenta si están en mayúsculas o minúsculas. |
| Variables que tengan etiquetas HTML. | Se usa una expresión regular para detectar etiquetas HTML que exista a una de las variables. |

## **Web Crawling / Spidering**

Para este caso se tomará como Crawling o spidering-bot cuando en un determinado lapso de tiempo se realice en promedio cierta cantidad de peticiones por segundo desde una misma dirección IP con el mismo User-Agent. La razón de obtener un promedio es porque algunas herramientas tardan en procesar las peticiones que van a lanzar al servidor, además se toma en consideración de que una misma dirección IP puede hacer distintas solicitudes con distintos User-Agent, por eso se considera como Crawler que una IP solicite muchos recursos y en cada petición use el mismo UserAgent.

Para identificar una herramienta de tipo Crawler se necesitan los siguientes elementos:

|  |  |
| --- | --- |
| Elementos para la detección para de herramientas Web Crawling | |
| Elemento | **Descripción** |
| Multiples peticiones en un lapso de 10 segundos. | Se necesita que existan muchas peticiones en un lapso de 10 segundos, sin embargo este valor se puede modificar por medio del archivo de configuración en la variable *duracionSeg*. |
| IP y User-Agent. | Para que sea detectada una herramienta, se necesita que las peticiones de una IP sean provenientes del mismo User-Agent |
| Promedio de peticiones realizadas en un segundo. | Se calcula el promedio de las peticiones realizadas y después se compara con una variable que se encuentra en el archivo de configuración, dicha variable se llama *frecPromPeticionSeg* y por defecto tiene el valor de 3 segundos. |
| Petición de distintos recursos. | El objetivo de un Crawler es el de elaborar un árbol del sitio, por lo tanto se verificará que no se solicite el mismo recurso en todas las peticiones. |
| Nombre del User-Agent correspondiente a una herramienta conocida para indexar sitios Web. | Para distinguir entre un Crawler y un bot se apoya del User-Agent. |

Como se mencionó anteriormente en el capítulo *2. Análisis de* bitácoras se requiere una estructura de datos que facilita el análisis, la cual se muestra a continuación:

Hash: Marca de tiempo hechas por Acunetix

Llave: {22/Sep/2016:16:50:21}

Arreglo: Recurso[0] = /robots.txt

Recurso[1] = /index.html

Recurso[2] = /images/

Recurso[3] = /images/algo.png

Llave: {22/Sep/2016:16:50:22}

Llave: {Acunetix}

Llave: {9.9.9.9}

Hash: IP

Arreglo: Recurso[0] = /scripts/

Recurso[1] = / scripts/setup

Recurso[2] = /scripts/algo

Recurso[3] = /admin/login.php

Hash: User-Agent

Llave: {22/Sep/2016:16:50:22}

Arreglo: Recurso[0] =

/admin/login.php

Llave: {Chrome}

Hash: Marca de tiempo hechas por Chrome

Existe un hash (cuadro azul) cuya llave es la dirección IP (9.9.9.9) el cual es un elemento único, el valor que regresa es la referencia de otro hash (cuadro amarillo) en donde las llaves ahora serán los User-Agent( Acunetix, Chrome, etc.) asociados a una dirección IP determinada, el valor que asociado es la referencia a otro hash (cuadro verde) en donde se tiene la marca de tiempo como llave, el cual es referencia a un arreglo que contiene los elementos solicitados durante ese instante.

**No**

**Si**

**No**

**Si**

**Si**

**No**

**No**

**Si**

No se reporta un Crawler o Bot.

Se reporta un Bot.

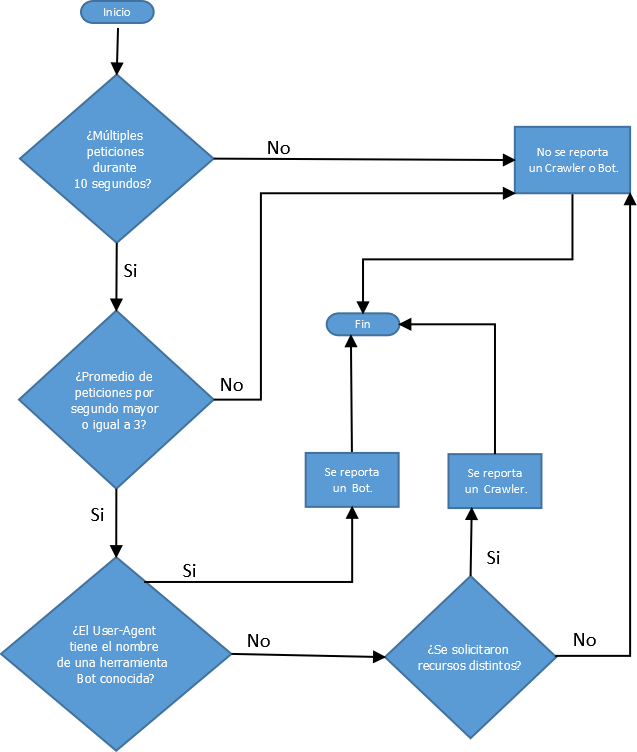
Se reporta un Crawler.

¿Se solicitaron recursos distintos?

¿El User-Agent tiene el nombre de una herramienta Bot conocida?

¿Promedio de peticiones por segundo mayor o igual a 3?

¿Múltiples peticiones durante 10 segundos?



## **Path Traversal**

Para este caso solo se detectarán aquellos intentos que se realicen en un sistema operativo derivado de UNIX, en este submódulo se analizan las variables y los directorios, teniendo dos listas negras las cuales se encuentran en el Anexo A.

El siguiente diagrama de flujo muestra el proceso del análisis:

¿Valor contiene palabra lista negra de recursos SIN variables?

¿Recurso solicitado tiene variables?

No hay Path Traversal.

No hay Path Traversal.

**Si**

**No**

Hay Path Traversal.

Se correlaciona el evento con la bitácora error.log

Hay Path Traversal.

Se correlaciona el evento con la bitácora error.log

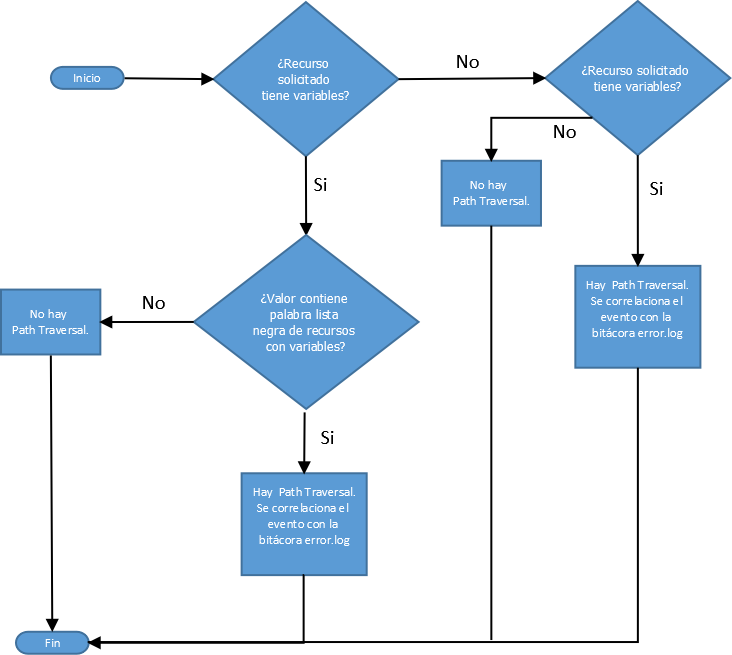
**No**

**Si**

**Si**

¿Valor contiene palabra lista negra de recursos con variables?

**No**



Para la detección de un evento de tipo Path Traversal se considera la lo siguiente:

|  |  |
| --- | --- |
| Elementos para la detección para de herramientas Web Crawling | |
| Elemento | **Descripción** |
| Recuso solicitado con variables cuyos valores estén en una lista negra | En el anexo A apartado 10.5 Lista negra para Path Traversal con variables en la URL se muestra la lista que se usa, en dicha lista además de buscar coincidencias con los caracteres ../ se verifica que no exista la solicitud de archivos comunes, por ejemplo /etc/passwd |
| Directorios solicitados con los caracteres ../ | Para este caso el atacante intenta de ejecutar este ataque por medio de los directorios de la URL, en esta lista solo se toma en cuenta los caracteres ../ los cuales pueden ir codificados en UNICODE a dos bytes, no se incluyen nombres de directorios para evitar falsos positivos. |

## **Defacement**

Para este análisis solo se verificará los métodos PUT y DELETE ya que es la única forma de detectar un defacement a través de las bitácoras, lo ideal sería verificar la integridad de los archivos que sean parte del sitio web pero no está dentro alcance del proyecto. A continuación se muestra el diagrama de flujo de este submódulo:

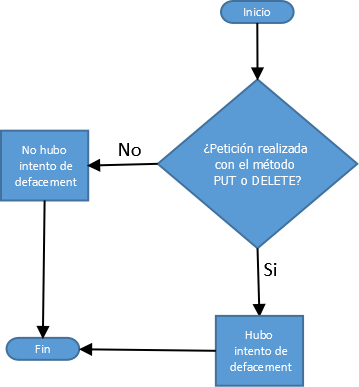
Hubo intento de defacement

No hubo intento de defacement

**Si**

¿Petición realizada con el método PUT o DELETE?

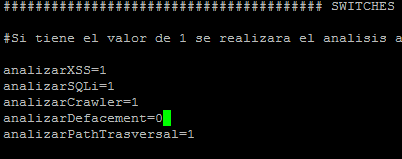
**No**



Como elemento de detección para este ataque se considera lo siguiente:

|  |  |
| --- | --- |
| Elemento para la detección para de herramientas Web Crawling | |
| Elemento | **Descripción** |
| Método HTTP | Solo se verifica el uso de los métodos PUT y DELETE |

Cabe señalar que si el servidor Web soporta los métodos HTTP PUT y DELETE o si cuenta con alguna aplicación como WebDav, podrían generarse falsos positivos, para este caso se recomienda desactivar el análisis para este ataque, para ello se realiza modificando el archivo *config.conf* en donde se encuentra una variable con el nombre *analizarDefacement* cuyo valor deberá de ser 0 (por defecto el valor es 1).



# **Análisis de bitácoras de ModSecurity**

Para el segundo módulo se tiene un script en Perl cuyo nombre es *modSecurity.pl*, la herramienta monitorea las alertas generadas por ModSecurity siempre y cuando este en modo *“Detection Only”*, esta verificación se realiza desde el script en Python llamado *Proyecto.py*. Cuando ModSecurity determina que hay un ataque se realiza una puntuación para concluir que tipo de ataque se ejecutó y esto se queda registrado en el archivo error.log en el servidor WAF, además se genera una entrada en el archivo modsec\_audit.log, ahí se encuentra el User-Agent y el payload. Cuando ModSecurity registra en el error.log la puntuación se realiza en el campo de msg de la siguiente forma:

msg "Inbound Anomaly Score Exceeded (Total Inbound Score: 48, SQLi=7, XSS=35): IE XSS Filters - Attack Detected."]

En todas las puntuaciones siempre usa la cadena “*Inbound Anomaly Score Exceeded*” por lo tanto servirá como referencia para saber si hubo un ataque o no, y de ahí se tomada la una id la cual es única, del archivo error.log se obtiene la siguiente información:

* Marca de tiempo.
* Dirección IP.
* Tipo de Ataque.
* Hostname o dominio.
* ID.

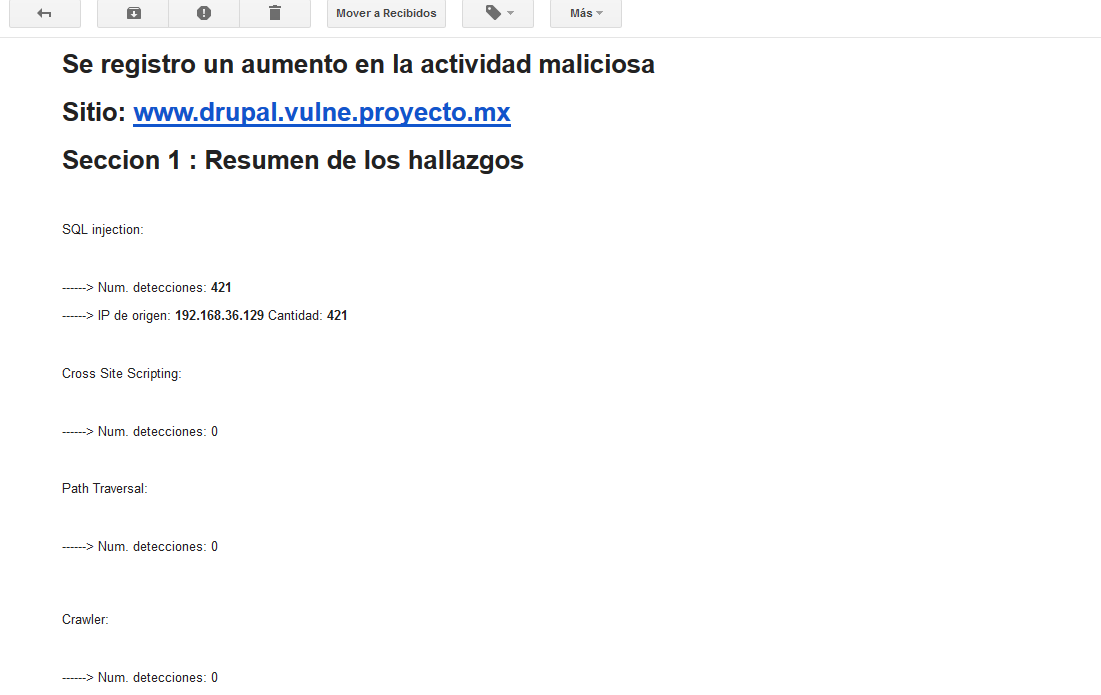
Una vez que se detecta una anomalía se consulta el archivo modsec\_audit.log y se correlaciona con el archivo error.log por medio de la ID, de la bitácora modsec\_audit.log se obtiene lo siguiente:

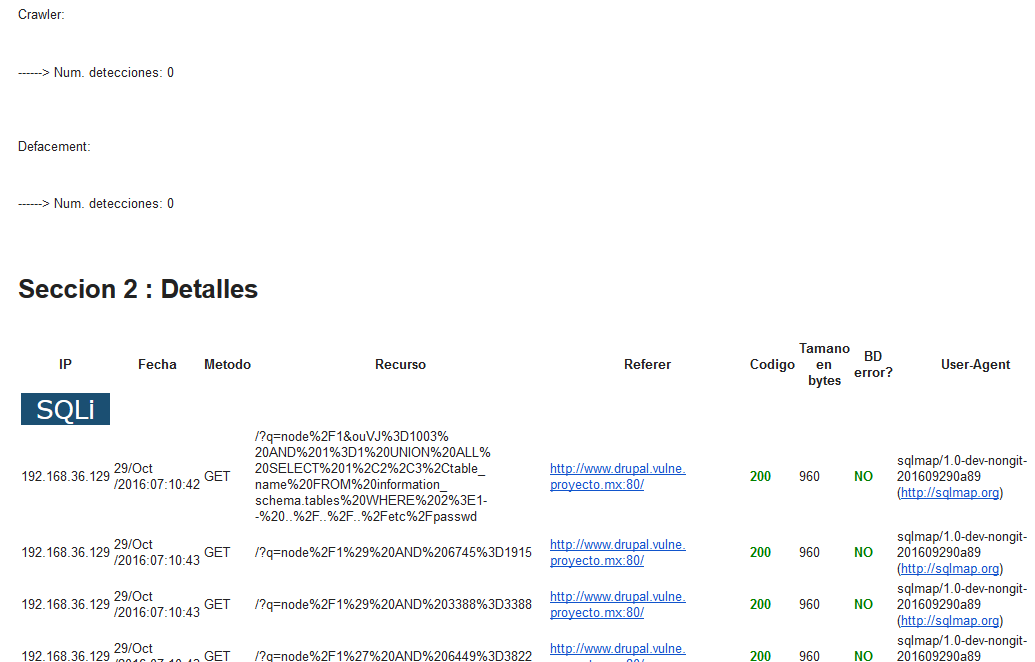
* Recurso solicitado (payload).
* User-Agent.

Ya que se obtiene

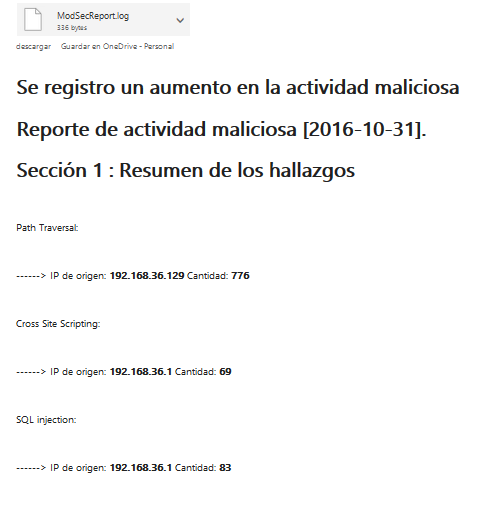
# **Reporte de hallazgos**

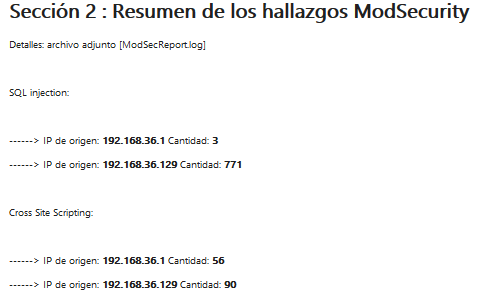
Una vez que se detecte algún ataque, se enviará un correo electrónico cada 15 minutos, a continuación se muestra un ejemplo:

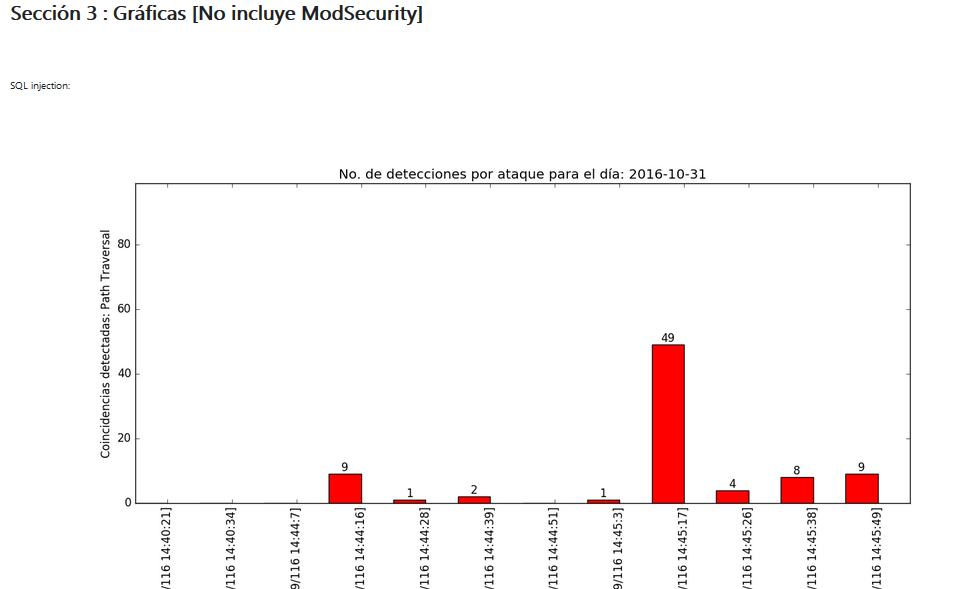




Por otra parte se enviará un correo electrónico diario con un reporte de los ataques detectados, a continuación se muestra un ejemplo:







Por otra parte se necesita de un correo de origen con su respectiva contraseña y uno de destino, en el archivo config.conf se define por medio de las variables de forma respectiva: mailFROM, passMail y mailTo.

# **Consideraciones previas y requisitos**

* La herramienta solo analizará ataques dirigidos un servidor Web Apache, un servidor de base de datos PostgreSQL los cuales cuentan con un servidor WAF (ModSecurity).
* En el servidor WAF será la instancia en donde vivirá la aplicación, la cual necesita de algunas bibliotecas, paquetes y otras dependencias las cuales se listan a continuación y se agregan al momento de la instalación:
  + Software instalado:
    - Python 2.7
    - Perl versión 5
    - OpenSSH Server
  + Bibliotecas necesarias:
    - Python:
      * Spur
      * Pysftp
      * Matplotlib
      * Shutil
      * Termcolor
      * Pandas
      * Numpy
      * Reportlab
    - Perl:
      * MIME::Base64
      * URI:Encode
      * Date::Parse
* Espacio en disco: 2 MB para la instalación de la aplicación, 500 MB para la instalación de dependencias y más de 1GB para almacenar las bitácoras que se extraen de los servidores.
* Al realizar la conexión SSH con los servidores Web y de base de datos se debe de elegir un usuario que tenga permisos de lectura en las bitácoras.
* En caso de que dicho usuario sea eliminado se necesitará copiar el par de llaves.
* Se deberá de personalizar un archivo de configuración para poder indicar lo siguiente:
  + - Analizar un ataque en específico (opcional).
    - Datos para calibrar el Crawler (opcional).
    - Crear un archivo por cada sitio instalado en el servidor web con sus respectivas rutas de las bitácoras (access.log y error.log).
* Se puede crear un usuario o usar uno que haya sido creado anteriormente en el servidor WAF para poder ejecutar la aplicación, el script de instalación installServer.sh solicita un nombre de usuario que exista o que se desea crear.
* La fecha y hora de todos los servidores deben de estar sincronizadas, como máximo debe de haber una diferencia de tres segundos (en el archivo de configuración se puede indicar el máximo tiempo de diferencia), de lo contrario la correlación de eventos fallará. De manera opcional al momento de instalar la aplicación, se puede sincronizar la fecha y hora en todos los servidores, para ello se requiere de un usuario con la capacidad de hacer esta acción por medio del comando date, para ello se puede apoyarse de sudo.
* No se deben de registrar las direcciones IPv6 en las bitácoras.
* Todos los logs deberán de registrar la marca de tiempo hasta los segundos.
* Tener instalado un servidor Web Apache 2.2.x
* Tener instalado un servidor de Base de datos PostgreSQL 9.1
* Tener un servidor con ModSecurity versión 2.9.
* Todos los servidores tienen instalado un sistema operativo Debian 7
* En todos los servidores se requiere mínimo 512 MB de memoria RAM.
* Se requiere tener acceso como root en el servidor WAF.
* El servidor WAF debe tener conexión a internet.
* Se requiere de un usuario para tener acceso a las bitácoras de los servidores (Web y base de datos).
* Se necesita realizar conexiones SSH del servidor WAF hacia los dos servidores.
* Se recomienda usar los siguientes repositorios:
* deb http://ftp.mx.debian.org/debian/ wheezy main contrib non-free
* deb-src http://ftp.mx.debian.org/debian/ wheezy main contrib non-free
* deb http://security.debian.org/ wheezy/updates main contrib non-free
* deb-src http://security.debian.org/ wheezy/updates main contrib non-free
* deb http://ftp.mx.debian.org/debian/ wheezy-updates main contrib non-free
* deb-src http://ftp.mx.debian.org/debian/ wheezy-updates main contrib non-free

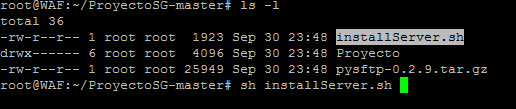
# **Instalación**

La herramienta se encuentra disponible en un repositorio git, cuyo enlace es el siguiente:

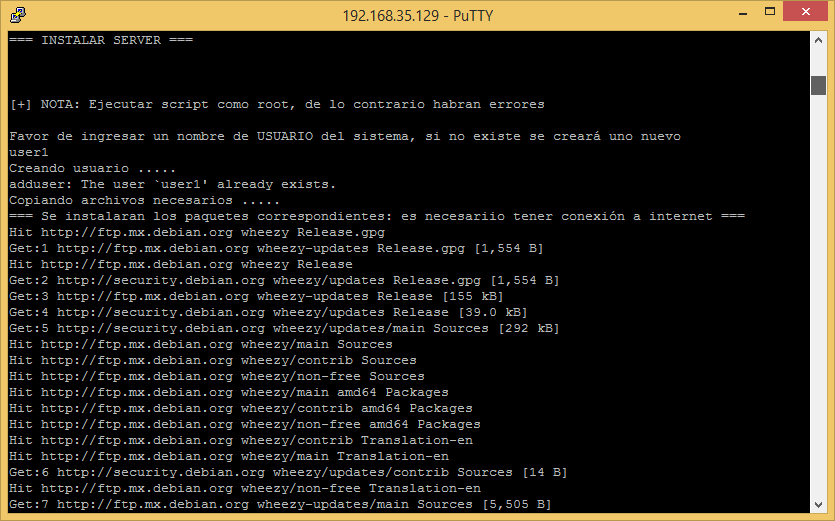
https://github.com/JuanPBSI/ProyectoSG

Para la instalación se divide en dos partes, la primera se realiza por medio de un script installServer.sh el cual va a instalar algunas dependencias, bibliotecas, se copian los componentes de la aplicación y además se asignará un usuario para ejecutar la herramienta. La segunda parte se crean un par de llaves para realizar las conexiones por SSH y se guardan algunos datos como direcciones IP y nombres de usuarios.

Iniciamos sesión como root en el servidor WAF, descargamos la herramienta, luego nos ubicamos en el directorio del proyecto y ejecutamos el script installServer.sh:

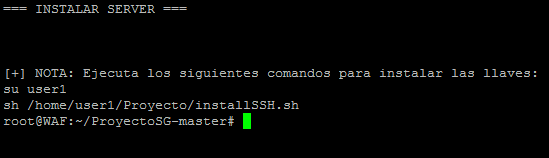


En seguida nos solicitará que escojamos un usuario, este puede ser uno que ya exista o de lo contrario este se creará, para este caso se usará el usuario *user1*:

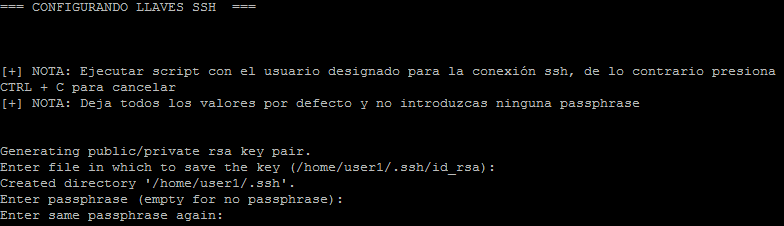


A partir de este momento se inicia la instalación de las dependencias, por lo tanto es posible que tarde en ejecutarse el script, dependiendo del ancho de banda del servicio de internet, además es probable que solicite alguna autorización para instalar algún paquete, en donde hay que aceptar toda solicitud.

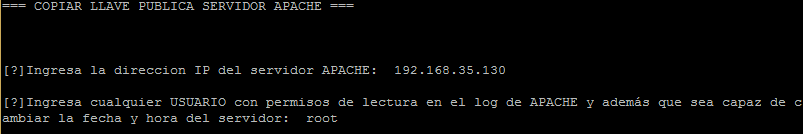
Después de instalar las dependencias se tendrá que ejecutar los comandos que aparecen a continuación para proseguir con la segunda parte de la instalación:



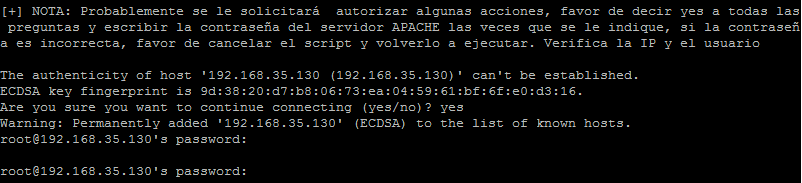
Luego nos pedirá que no introduzcamos una contraseña, por lo tanto solo apretamos *ENTER*



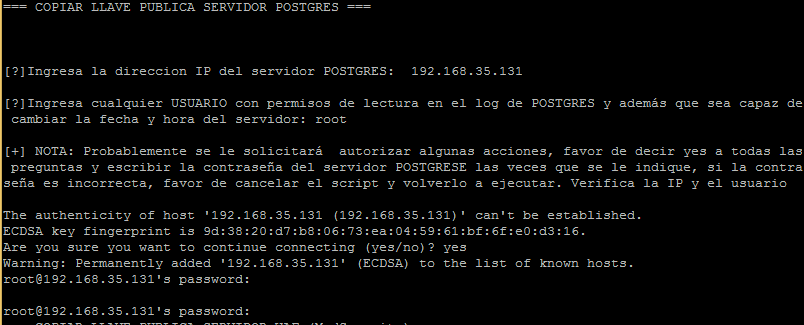
Posteriormente nos solicitará la dirección IP del servidor Web y su respectivo usuario:



En caso de que nunca se haya hecho una conexión nos solicitará una autorización para agregar la llave a la lisa de host conocidos, le indicamos que si escribiendo *yes* y luego ingresamos la contraseña del usuario que seleccionamos anteriormente las veces que lo solicite:



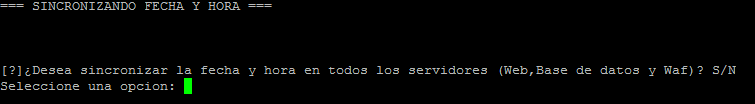
A continuación se realiza nuevamente el procedimiento anterior en los servidores PostgreSQL y WAF:



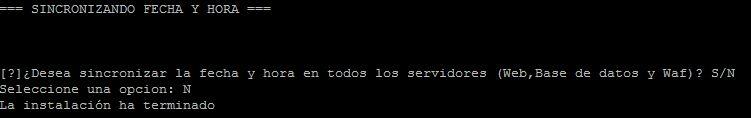


Para el caso del servidor WAF debemos de seleccionar un usuario que tenga permisos de lectura en las bitácoras de ModSecurity y las de Apache.

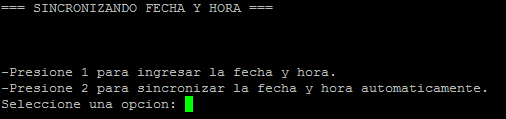
Posteriormente de manera opcional se sincroniza la fecha y hora en los tres servidores, nos aparecerá la siguiente pantalla para aceptar dicha acción:



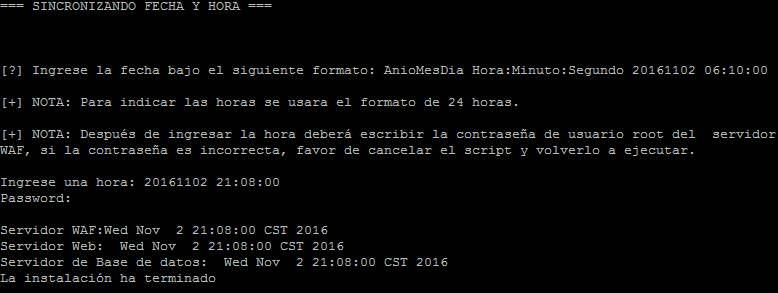
En caso de que no se realice la sincronización, la instalación habrá terminado:



Si se aceptó la sincronización de la fecha y hora se muestra lo siguiente:

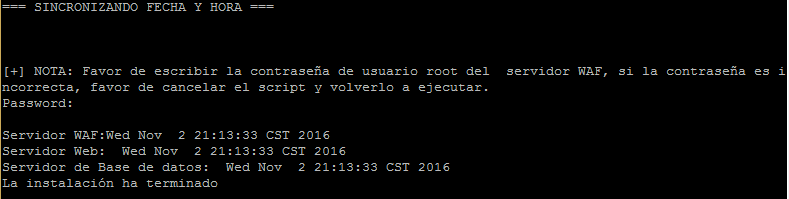


Si elegimos la opción 1 podremos indicar una fecha y hora en específico:



La fecha y hora que se ingresa debe de seguir el formato, de lo contrario se tendrá que ejecutar de nuevo el sript.

Para el caso de la opción 2 se obtiene la hora de ese momento del servidor WAF y se replica en los demás servidores.



# **Configuración**

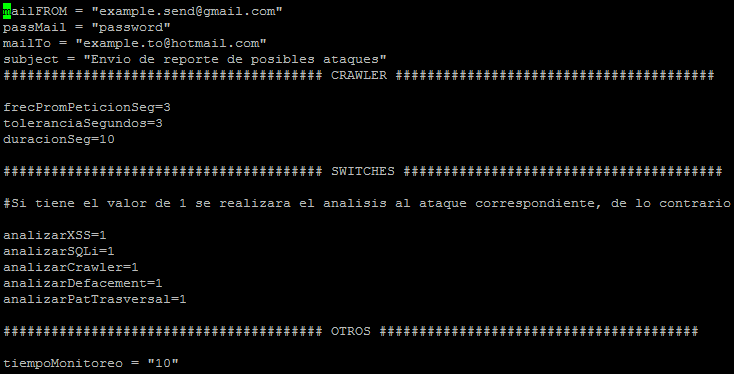


## **Archivo de configuración**

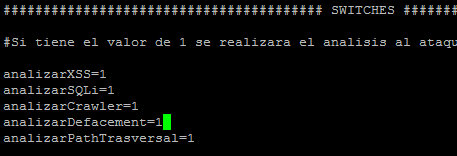
Para configurar la aplicación solo se requiere modificar un archivo de configuración con cualquier editor de texto (vi, nano, etc.) dentro del directorio de Proyecto existe un archivo llamado config.conf:



En dicho archivo solo se puede modificar lo siguiente, de lo contrario la aplicación dejará de ejecutarse correctamente:



* mailFROM: Esta variable sirve para indicar la cuenta de correo electrónico que se usará para enviar los reportes. El correo debe de estar entre comillas dobles como se muestra en la imagen anterior.
* passMail: Se define la contraseña del correo que se usará para enviar los reportes. La contraseña debe de estar entre comillas dobles como se muestra en la imagen anterior.
* mailTo: Se establece la cuenta de correo destinatario, es decir, la cuenta que recibirá los reportes. El correo debe de estar entre comillas dobles como se muestra en la imagen anterior.
* subject: Es el asunto del correo que se mandará. El asunto debe de estar entre comillas dobles como se muestra en la imagen anterior.



* analizarXSS: Es una bandera para indicar si se desea analizar ataques de Cross Site Scripting, si vale 1 se ejecutará el análisis, cualquier valor distinto se omitirá el análisis.
* analizarSQLi: Es una bandera para indicar si se desea analizar ataques de SQL injection, si vale 1 se ejecutará el análisis, cualquier valor distinto se omitirá el análisis.
* analizarCrawler: Es una bandera para indicar si se desea analizar si hubieron Crawlers, si vale 1 se ejecutará el análisis, cualquier valor distinto se omitirá el análisis.
* analizarDefacement: Es una bandera para indicar si se desea analizar ataques de Defacement, si vale 1 se ejecutará el análisis, cualquier valor distinto se omitirá el análisis.
* analizarPatTrasversal: Es una bandera para indicar si se desea analizar ataques de Path Traversal, si vale 1 se ejecutará el análisis, cualquier valor distinto se omitirá el análisis.

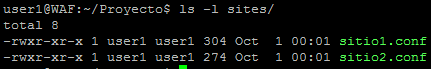


Finalmente la variable *tiempoMonitoreo* sirve para indicar cada cuanto tiempo se va a ejecutar la aplicación, dicho valor está expresada en segundos, cabe mencionar que este valor no debe ser menor al valor que se tiene en la variable *duracionSeg* el cual sirve para el análisis del Crawler, por defecto se tiene 30 segundos, no se recomienda modificar este valor ya que puede presentar problemas para detectar algunos ataques.

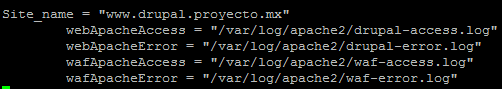


## **Dar de alta los sitios Web instalados**

Para definir todos los sitios que se tienen en el servidor Web se tiene que definir por medio del archivo de configuración que se ubica en el directorio *sites*:



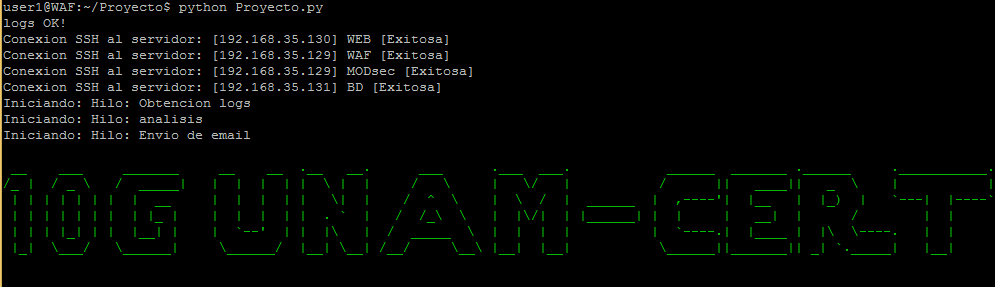
Por defecto se tiene un archivo llamado sitio1.conf, se debe de tener un archivo por cada sitio instalado, en caso de tener más de un sitio se debe de copiar el archivo *sitio1.conf* con otro nombre distinto en ese mismo directorio. Para cada archivo de configuración debemos de modificar su contenido con cualquier editor de texto, a continuación veremos el contenido del archivo *sitio1.conf*:



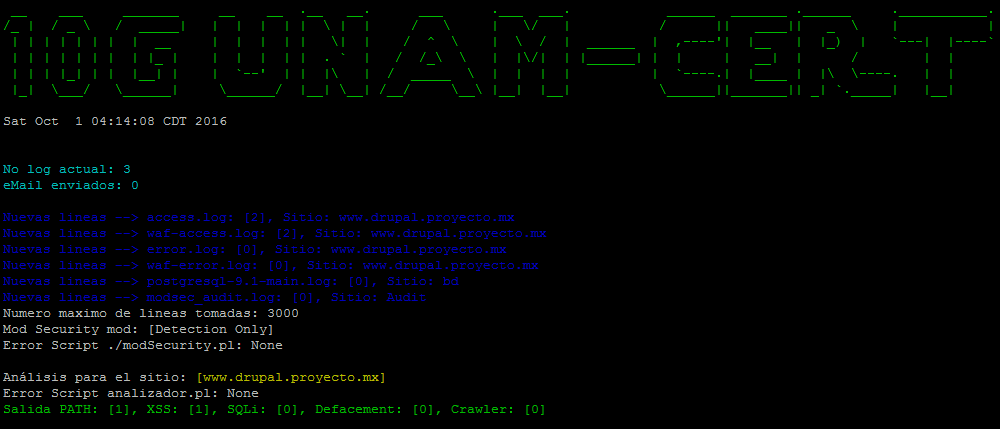
* webApacheAccess: Con esta variable indicamos la ruta del archivo access.log de nuestro servidor Web Apache. La ruta se escribe entre comillas.
* webApacheError: Definimos la ubicación del archivo error.log de nuestro servidor Web Apache. La ubicación se escribe entre comillas.
* wafApacheAccess: Se indica la ruta del archivo access.log de nuestro servidor WAF ModSecurity. La ruta se escribe entre comillas.
* wafApacheError: Se indica la ruta del archivo error.log de nuestro servidor WAF ModSecurity. La ruta se escribe entre comillas.

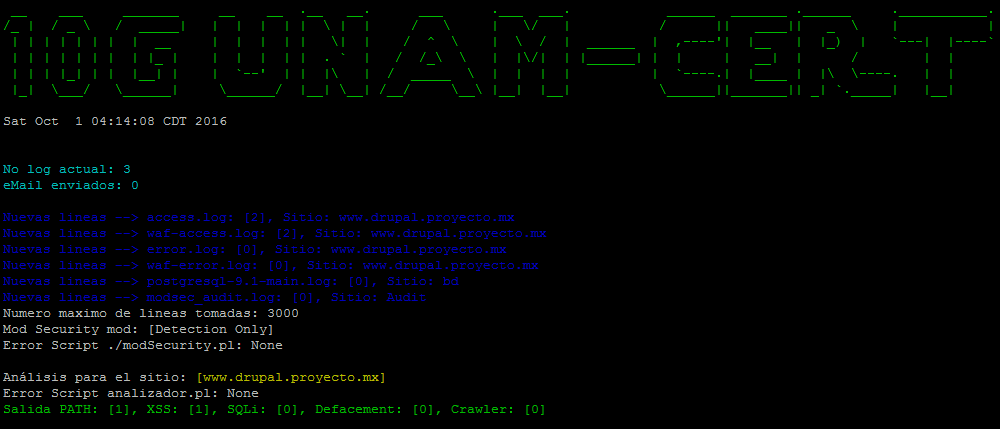
# **Modo de uso**

Nos situamos en el directorio *Proyecto* y ejecutamos el script *Proyecto.py*:



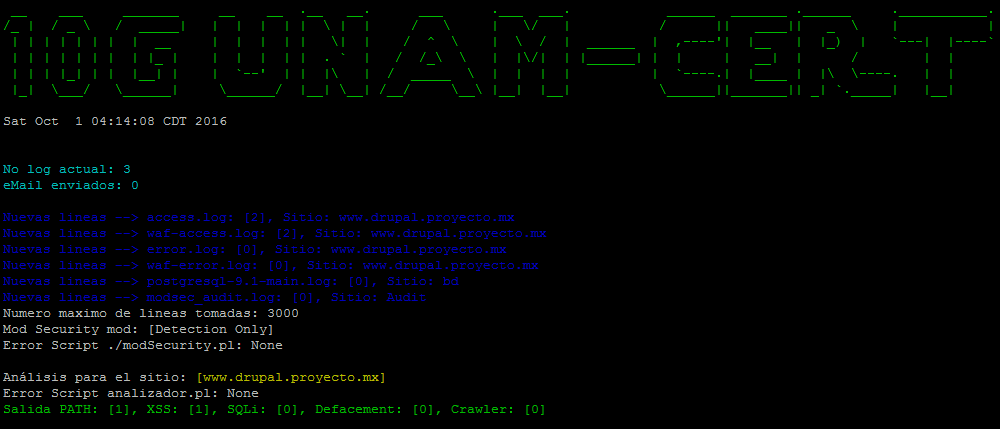
Una vez que aparezca en pantalla las mismas leyendas como en la imagen anterior significa que hemos instalado y configurado correctamente la herramienta, a continuación veremos un ejemplo de algunos ataques:

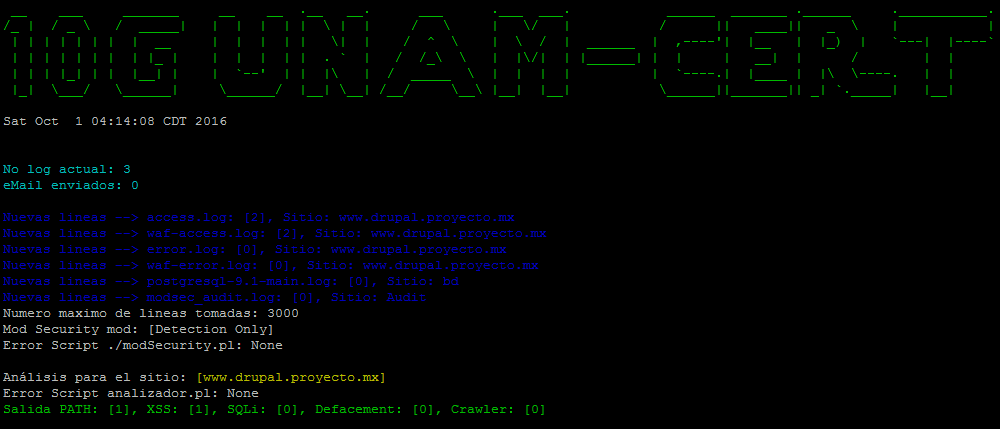




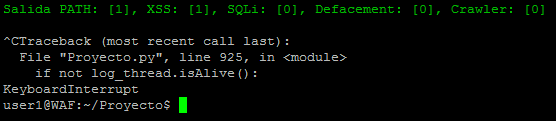
En la imagen anterior en letras verdes se aprecia los ataques que se presentaron, en este caso ocurrió uno de Path Traversal y otro de XSS para el sitio de *www.drupal.proyecto.mx* (letras amarillas), para obtener mayor detalle sobre los ataques se tiene que consultar el correo electrónico.

En los apartados de *Error Script* aparece los errores que sucedieron al hacer el análisis en este caso aparece la palabra *None*, el cual significa que el análisis fue ejecutado correctamente sin errores.





Finalmente cuando se necesite dejar de ejecutar la herramienta solo basta con apretar la secuencia de botones CTRL + C y mostrará un mensaje similar como el de la siguiente imagen:



# **Mejoras a futuro**

* Compatibilidad con servidores IIS.
* Compatibilidad con servidores SQL.
* Capacidad de determinar el tipo de ataque efectuado sobre los servidores, ejemplos.
  + SQL blind, Union, TimeBased, ErroBased, etc.
  + Cross Site Scripting: reflejado, almacenado, DOM based, etc.
* Compatibilidad con diversos formatos de las bitácoras .
* Incluir detección de XSS permanente.
* Anexar la detección de ataques de tipo Remote File Inclusion.
* Mejorar la clasificación de las alertas detectadas por ModSecurity.
* Verificar de manera periódica que la hora y fecha estén sincronizados en todos los servidores.
* Capacidad de detectar ataques en la red interna ( correlación exacta de eventos entre las bitácoras del servidor Web y el servidor WAF)

# **Anexo A**

A continuación se muestra las listas negras que se usan, las cuales pueden ser modificadas con cualquier editor de texto.

* 1. **Lista negra para detección de herramientas**

La siguiente lista se encuentra en un archivo llamado herramientas.txt en el directorio listas, una vez instalada la herramienta se tendrán las siguientes palabras:

* Acunetix
* Nikto
* sqlmap
* Nessus
* Vega
* Zed Attack Proxy
* OpenVAS
* Burp
* wget
* curl
* Skipfish
* AppScan
* Wikto
  1. **Lista negra para SQL injection**

La siguiente lista se encuentra en el script *analizador.pl*:

* ALTER
* CREATE
* DELETE
* DROP
* EXEC
* INSERT INTO
* MERGE
* SELECT
* UPDATE
* UNION
* OR
* AND
* ORDER BY
* WHERE
* HAVING
  1. **Lista negra para Cross Site Scripting**
  + script
  + src
  + onerror
  + onclick
  + onmouseover
  + iframe
  + eval
  + base64
  + fromCharCode
  + svg
  + <!--
  + -->
  + \/\/
  + \/\\*
  + \\*\/

Esta lista negra se encuentra en el directorio listas en el archivo XSS\_blacklist.txt, cabe señalar que se pueden agregar más palabras teniendo en consideración que algunos caracteres se deben de escapar con el carácter \, esto es debido a que se puede interpretar en un contexto distinto, los caracteres que deben de escaparse son:

* \*
* .
* [
* ]
* {
* }
* /
* ^
* $
* ?
* +

## **Lista negra para Bots**

A continuación se muestra la lista negra del User-Agent usado en las herramientas más comunes de los buscadores para indexar los sitios:

* Googlebot
* Bingbot
* Slurp
* DuckDuckBot
* Baiduspider
* YandexBot
* Sogou
* Exabot
* archive.org\_bot
* Teoma

## **Lista negra para Path Traversal con variables en la URL**

* ../
* ..\
* .%c0%af
* .%e0%80%af
* %c0%ae/
* %e0%80%c0%ae/
* %c0%ae%c0%af
* %e0%80%c0%ae%c0%af
* %c0%ae%e0%80%af
* %e0%80%c0%ae%e0%80%af
* .%c1%9c
* .%e0%80%9c
* %c0%ae\
* %e0%80%c0%ae\
* %c0%ae%c1%9c
* %e0%80%c0%ae%c1%9c
* %c0%ae%e0%80%9c
* %e0%80%c0%e0%80%9c
* etc/passwd
* etc/apache2
* etc/group
* etc/hosts
* etc/
* tmp/
* .htaccess
* home/

## **Lista negra para Path Traversal sin variables en la URL**

* ../
* ..\
* .%c0%af
* .%e0%80%af
* %c0%ae/
* %e0%80%c0%ae/
* %c0%ae%c0%af
* %e0%80%c0%ae%c0%af
* %c0%ae%e0%80%af
* %e0%80%c0%ae%e0%80%af
* .%c1%9c
* .%e0%80%9c
* %c0%ae\
* %e0%80%c0%ae\
* %c0%ae%c1%9c
* %e0%80%c0%ae%c1%9c
* %c0%ae%e0%80%9c
* %e0%80%c0%e0%80%9c

Algunos elementos son caracteres codificados en UNICODE a dos bytes, el cual no se decodifican en el script *analizador.pl*, por lo tanto se agregan en el diccionario. Prácticamente se busca que en la URL, en la sección de los directorios no existe la secuencia de caracteres ../, ..\ y sus respectivas equivalencias en la codificación en UNICODE con dos bytes.

# **Referencias**

Quan Bai, Gang Xiong, Yong Zhao, Longtao He. (2014). Analysis and Detection of Bogus Behavior in Web Crawler Measurement . Octubre 31, 2016, de Elsevier B.V. Sitio web: http://www.sciencedirect.com/science/article/pii/S1877050914005407

Dario Valentino Forte. (Desconocido). THE “ ART ” OF LOG CORRELATION Tools and Techniques for Correlating E vents and Log Files. Octubre 31, 2016, de Desconocida Sitio web: http://docplayer.net/4962416-The-art-of-log-correlation.html

Desconocido. (2014 ). Double Encoding. Octubre 31, 2016, de OWASP Sitio web: https://www.owasp.org/index.php/Double\_Encoding

Roger Meyer. (2008). Detecting Attacks on Web Applications from Log Files. Octubre 31, 2016, de SANS Institute Sitio web: https://www.sans.org/reading-room/whitepapers/logging/detecting-attacks-web-applications-log-files-2074

Issac Museong Kim. (2011). Using Web Application Firewall to detect and block common web application attacks. Octubre 31, 2016, de SANS Institute Sitio web: https://www.sans.org/reading-room/whitepapers/webservers/web-application-firewall-detect-block-common-web-application-attacks-33831

T. Berners-Lee, L. Masinter, L. Masinter, M. McCahill. (1994). Uniform Resource Locators (URL). Octubre 31, 2016, de IETF Sitio web: http://www.ietf.org/rfc/rfc1738.txt

Desconocido. (2012). Fun with data: URLs. Octubre 31, 2016, de Desconocida Sitio web: http://blog.kotowicz.net/2012/04/fun-with-data-urls.html

Desconocido. (2015). Path Traversal. Octubre 31, 2016, de OWASP Sitio web: https://www.owasp.org/index.php/Path\_Traversal

Desconocido. (2016). XSS Filter Evasion Cheat Sheet. Octubre 31, 2016, de OWASP Sitio web: https://www.owasp.org/index.php/XSS\_Filter\_Evasion\_Cheat\_Sheet

Desconocido. (2016). XSS (Cross Site Scripting) Prevention Cheat Sheet. Octubre 31, 2016, de OWASP Sitio web: https://www.owasp.org/index.php/XSS\_(Cross\_Site\_Scripting)\_Prevention\_Cheat\_Sheet

Desconocido. (2015). DOM Based XSS. Octubre 31, 2016, de OWASP Sitio web: https://www.owasp.org/index.php/DOM\_Based\_XSS

Bijender Singh Bish. (Desconocido). How to exploit HTTP methods – PUT and DELETE. Octubre 31, 2016, de HTTP Secure Sitio web: http://httpsecure.org/?works=how-to-exploit-http-methods-put-and-delete

Desconocido. (2012). ModSecurity 2 Data Formats . Octubre 31, 2016, de Trustwave Holdings, Inc. Sitio web: https://github.com/SpiderLabs/ModSecurity/wiki/ModSecurity-2-Data-Formats

Etel Sverdlov. (2012). How To Set Up SSH Keys. Octubre 31, 2016, de DigitalOcean Sitio web: https://www.digitalocean.com/community/tutorials/how-to-set-up-ssh-keys--2