Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de las Telecomunicaciones

Sistema de monitorización y correlación de eventos en gestión de redes

Autor: Antonio Cuesta García
Tutor: Antonio José Estepa Alonso

Departamento de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2016

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de las Telecomunicación

Sistema de monitorización y correlación de eventos en gestión de redes

Autor:
Antonio Cuesta García

Tutor:
Antonio José Estepa Alonso
Profesor titular

Departamento de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2016

Trabajo Fin de Grado: Sistema de monitorización y correlación de eventos en gestión de redes

Autor: Antonio Cuesta García

Tutor: Antonio José Estepa Alonso

El tribunal nombrado para juzgar el Trabajo arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2016

El Secretario del Tribunal

A todas las personas que lo han hecho posible. En especial a mis padres.

Agradecimientos

En este punto, al finalizar ya mi carrera me doy cuenta de lo rápido que han pasado estos 4 años de la carrera, los más rápidos de mi vida. Aunque, a la vez han sido los más duros con muchas tristezas, han sido los más bonitos que he podido vivir juntos con mis amigos de la facultad, mis compañeros del Colegio Mayor y con el apoyo incondicional de toda mi familia. Todos ellos se merecen un gran agradecimiento por mi parte.

Gracias a mis padres, Carmen y Martín, por haber hecho posible todo esto en el que han contribuido con un gran esfuerzo.

Gracias a mi hermano, Martín, que me ha apoyado cuando mis padres no me llegaban a comprender del todo.

Gracias a mis compañeros de clase: Pablo, Juan Emilio, Ismael, Bea, Carmen "Carmeli" … Gracias por hacerme este camino mucho más fácil.

Gracias a todo mi Colegio Mayor San Juan Bosco, en especial a mi grupo Cuetara, no podía faltar en este agradecimiento el cual me ha ayudado tanto en el plano académico como en el espiritual, que tienen que llevarse de la mano.

Por último, gracias a mi tutor Antonio Estepa, que, aunque haya habido momentos en que ni iba a visitarle a su despacho, me ha guiado en todo momento.

A todos vosotros,

GRACIAS.

Antonio Cuesta García

Estudiante del Grado en Ingeniería de las Tecnologías de Telecomunicación

Sevilla, 2016

Resumen

Este trabajo de fin de grado se centra en el análisis y comparación de tres sistemas de monitorización y correlación de eventos en la gestión de redes.

Comienza con una investigación sobres los correladores de eventos, para posteriormente definir los software usados para el desarrollo del mismo. Entre estos servicios se usan SEC, Nxlog y Prelude, que también son tema central de algunos apartados, ya que sobre ellos se sustenta el desarrollo de este trabajo. Posteriormente se definen los aspectos más técnicos del experimento realizado.

Por último, en los anexos, se explicará cómo se procede a la instalación y configuración de los tres sistemas.

El resultado obtenido al final de esta investigación tiene dos vertientes, si se quiere gestionar una red orientada tanto a las pequeñas como medianas empresas se puede hacer uso de estas herramientas, pero para empresas con necesidades de alta carga habría que dirigir la mirada a una versión comercial[1].

[1] Una versión comercial, es una versión de pago.

Abstract

This final degree work focuses on the analysis and comparison of three systems of monitoring and correlation of events in network management.

It begins with an investigation on the correlators of events, to further define the software used for development. These services are SEC, Nxlog and Prelude, which are also central theme of some sections, since it is based on them the development of this work. Then the more technical aspects of the experiment performed are defined.

Finally, in the annexes, it will explain how to proceed to the installation and configuration of the three systems.

The result obtained at the end of this research is twofold, if you want to manage both small and medium enterprises network can make use of these tools, but for companies with high load needs should turn our gaze to a comercial version.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 INTRODUCCCIÓN Y OBJETIVOS

If you can't explain it simply, you don't understand it well enough.

- Albert Einstein -

Cada vez tenemos infraestructuras de red más complejas en las que se compromete información tanto pública como privada. Existen unos archivos llamados logs en los cuales un SO, dispositivo de red, aplicación, etc, deja información para el administrador acerca de lo que ha ocurrido en el sistema.

1.1 Contexto

Todos y cada uno de los equipos existentes en una red (routers, switches, firewalls, servidores, …) generan una serie de eventos y los almacenan en los archivos de log o de registro para que los administradores de sistemas puedan consultarlos en caso de ser necesario. En ellos podemos encontrar:

- Intentos no autorizados
- Información del tráfico de red.
- Errores que se producen en los propios dispositivos.
- Los servidores web generan un evento cuando un cliente accede a alguna de las páginas web que alberga.

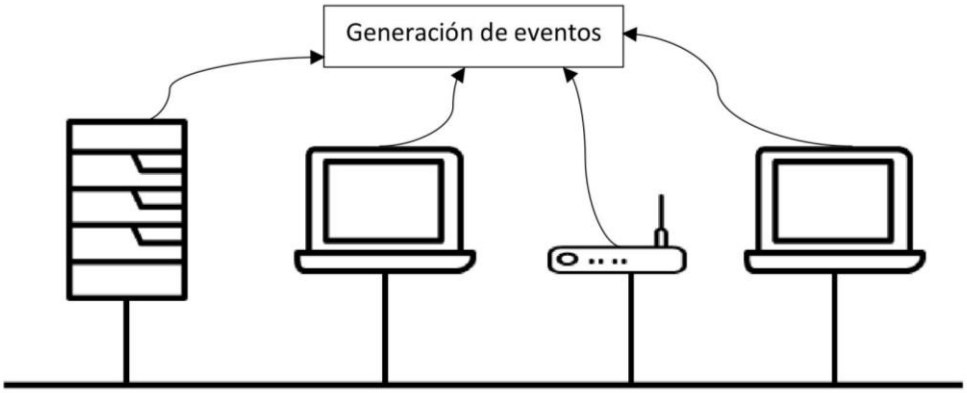


Figura 1-1. Generación de eventos

1.2 Problema

En la realidad, un problema podría generar muchos eventos, lo que conducen a problemas complejos que tienen que ser gestionados manualmente.

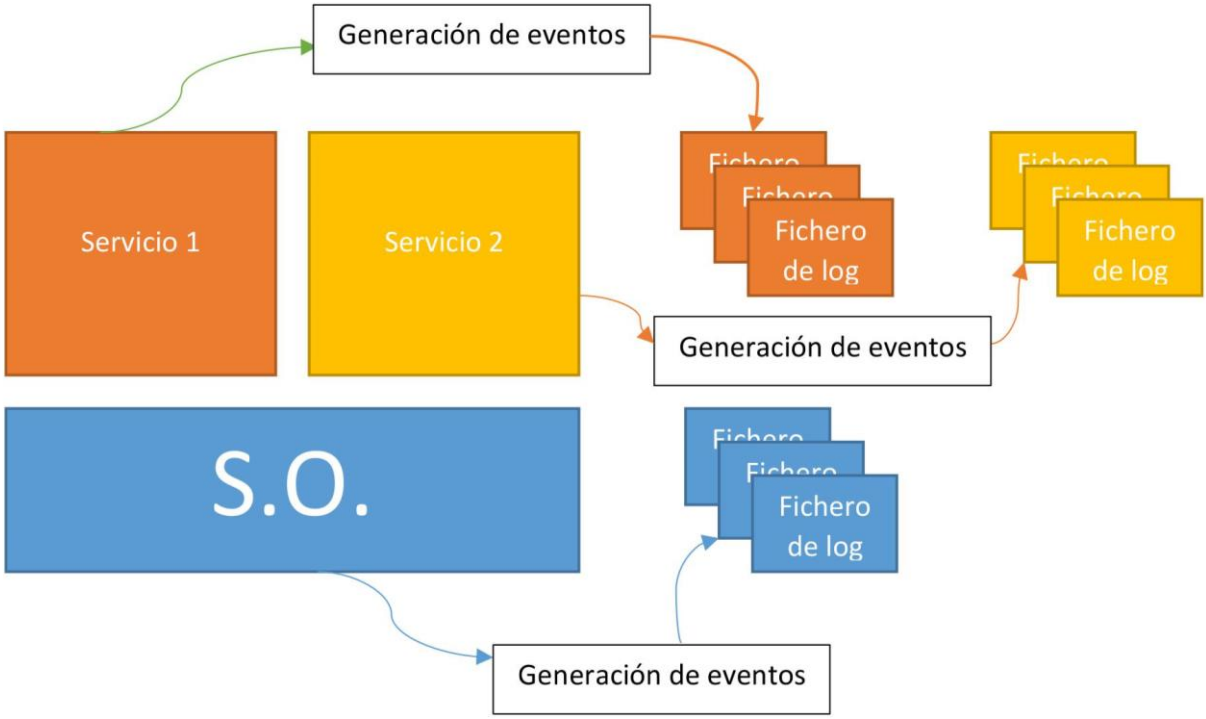


Figura 1-2. Arquitectura de generación de eventos

En general, en los sistemas y dispositivos se tiene muchos ficheros de logs con muchos eventos, lo cual lleva a un trabajo laborioso para su revisión manual por parte del gestor de la red.

1.3 Solución

Nos gustaría reducir el número de eventos y que fueran más significativos. Para ello, haremos uso de la correlación de eventos que nos ayudaran a reducir el número de eventos que deben ser examinados.

Actualmente, hay diversos motores de correlación, pero no son ampliamente conocidos ni por empresas ni por particulares. Los cuales ayudan a la seguridad interna y a llevar una gestión mucho más adecuada de la red.

1.4 Objetivos

La idea de este proyecto nace con la necesidad de dar una comparación entre tres motores de correlación: SEC, Nxlog y Prelude, para la elección final del usuario. En esta comparación se hará desde tres puntos de vista:

- Funcionalidades de las herramientas.
- Tiempos de respuesta, para comprobar la eficiencia de dichas herramientas.
- Puesta en marcha, en el que se engloba: su instalación, configuración y escritura de reglas.

1.5 Estructura y metodología del trabajo

En primer lugar, se realizará una investigación sobre los correladores de eventos, para tener un conocimiento y una visión general de ellos. Se investigará también las herramientas que formarán parte de este proyecto para aprender sobre su uso.

Una vez concluida la fase de investigación, se desarrollarán en un entorno de pruebas las configuraciones necesarias para que se cumplan los objetivos propuestos para el proyecto.

Y, terminaremos con una discusión de los sistemas y de los resultados obtenidos en la fase de desarrollo

2 LOGS

Information is the oil of the 21st century, and analytics is the combustion engine.

- Peter Sondergaard -

No podemos empezar este trabajo sin introducir previamente uno de los dos conceptos más importantes de este. Se trata del concepto de log, el cuál será repetido constantemente en esta memoria por ser la gestión de estos la finalidad de este trabajo.

Durante este capítulo hablaremos sobre lo que es un log y porque son tan importantes. Al acabar este capítulo el lector debe tener una visión general del mundo de los logs, que le servirá para entender el resto de la memoria. [1]

2.1 ¿Qué es un log?

Cuando hablamos de log nos referimos a los archivos que almacenan los eventos, líneas de log o líneas de evento, que son generados por: un dispositivo de red, aplicación, etc.

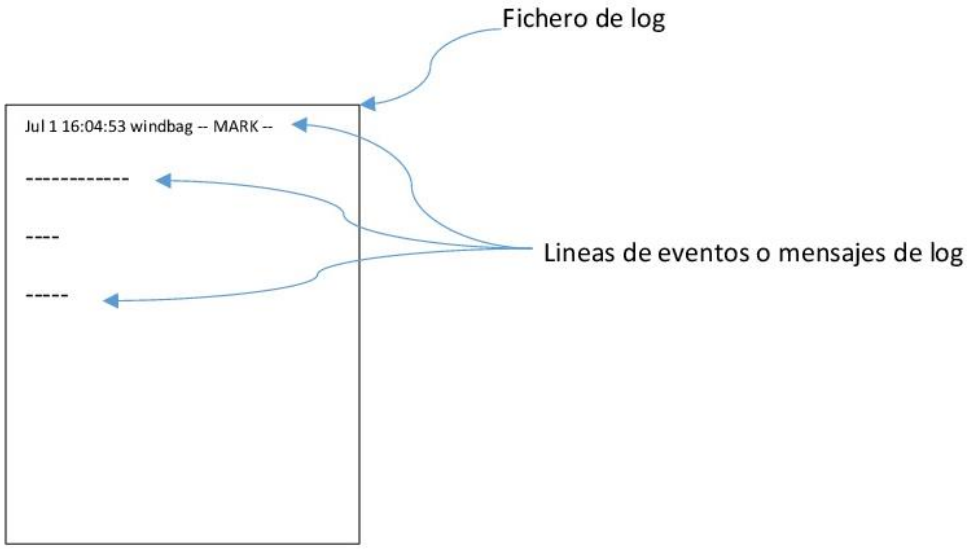


Figura 2-1. Fichero de log

Estos eventos pueden ser muy variados dependiendo de la fuente que lo genere. Por ejemplo, los sistemas UNIX generan un evento cada vez que alguien inicia sesión o cierra sesión con cualquier usuario, los controladores de APs generan un evento cuando algún equipo se conecta o desconecta de alguno de los puntos de acceso de la red, los servidores web generan un evento cuando un cliente accede a alguna de las páginas web que alberga.

Retomando los ejemplos anteriores, en el caso de los sistemas UNIX, el evento nos indicaría el usuario con el

que se ha iniciado o cerrado la sesión; en el caso del controlador, nos indica la dirección MAC de la estación que se ha conectado y a qué punto de acceso; en el caso del servidor web, nos indica que página ha sido accedida, desde que IP, con que navegador, etc.

2.1.1 Categorías

Cada log tiene diferentes categorías, dependiendo del evento que lo haya generado, las cuales son:

- *Debug:* son generados por las aplicaciones con la finalidad de ayudar a los desarrolladores a identificar problemas y depurar el código de dicha aplicación.
- *Informational:* este tipo de eventos están pensados para hacer saber a los usuarios y administradores que ha ocurrido algo sin gravedad. Por ejemplo, cuando un servidor se reinicia genera un evento. Esto en principio no tendría mayor importancia, a no ser que el reinicio se produjese en una situación en la que no debería, por ejemplo, cuando el servidor está en producción y no se está realizando mantenimiento. En este caso, entraría en acción la correlación de eventos, dados dos situaciones, ¿es correcto lo que sucedió?
- *Notice:* generados por eventos que no son frecuentes, pero no son considerados errores, por lo que su importancia, al igual que el tipo anterior, dependerá del contexto.
- *Warning:* estos eventos están pensados para situaciones donde falte algo que el sistema necesita, pero la ausencia de esto no tiene impacto sobre el correcto funcionamiento del sistema. También son usados para indicar que ocurrirá un error si no se realiza alguna acción. Un ejemplo de esto sería cuando se ejecuta una aplicación con un número incorrecto de argumentos, pero, aun así, la aplicación puede ejecutarse.
- *Error:* se usan para notificar los errores ocurridos en el sistema. Por ejemplo, un sistema operativo generará un evento de tipo error cuando una aplicación haya excedido su límite de almacenamiento y sus intentos de escritura están fallando.
- *Critical:* notifican algo que debe ser corregido, pero no es necesario que sea inmediatamente, por ejemplo, la pérdida de conexión con el ISP secundario mientras la conexión con el primario funciona correctamente.
- *Alert:* se usan para notificar algo que debe ser corregido inmediatamente, por ejemplo, la perdida de conexión con el ISP principal y secundario. Están relacionados también con el dominio de los dispositivos y sistemas de seguridad, que generan un log de este tipo cuando detectan conexiones maliciosas.
- *Emergency:* notifican eventos por los que el sistema ha quedado inservible. No deben ser usados por aplicaciones.

Cabe mencionar que el significado de estos niveles (exceptuando debug y emergency), son relativos a la aplicación que genera el log. Por ejemplo, no tiene la misma importancia un log de nivel warning generado por la aplicación de gestión de correo electrónico que un log del mismo nivel generado por el kernel de Linux.

2.2 ¿Por qué son importantes?

Los logs están infravalorados en muchos entornos empresariales. Tanto es así que a veces los logs son ignorados hasta el punto de que sólo se acude a ellos cuando queda poca memoria en el disco duro, y muchas veces se borran sin consultarlos previamente, sin tener en cuenta que estos logs podrían contener información que nos diga porqué el disco duro está lleno.

Esto es debido a que analizar los logs no es una tarea sencilla, más bien es una tarea que en la mayoría de los casos nos supone mucho trabajo. Los logs tienen multitud de formatos y tamaño, y a veces puede ser difícil extraer información de ellos (como veremos en la sección 2.4). A esto se suma que algunos entornos se acumulan varios GB de logs a la semana o incluso al día.

A pesar de esto, debemos recordar que los logs nos proporcionan información muy importante sobre lo que

está ocurriendo en nuestra red y que, gracias a los correladores de eventos, la terea de análisis de logs no es tan compleja.

A continuación, mencionaremos algunos de los casos de uso que nos proporcionan los logs e intentaremos ilustrarlos con ejemplos.

2.2.1 Administración de recursos

La información contenida en los logs nos permite determinar el estado de los equipos de nuestra red, los fallos que ocurren, tanto de software como de hardware, y que están haciendo los servicios que se están ejecutando en nuestros equipos.

Un ejemplo de esto lo encontramos cuando queremos ver si un equipo conectado a la red está funcionando correctamente o está bloqueado o apagado. Una forma de averiguarlo es haciendo uso del protocolo ICMP, pero esta técnica no es infalible. Hacer ping satisfactoriamente a un equipo conectado a la red te dice que la tarjeta de red está configurada correctamente y que está encendida, pero puede ser que el sistema esté bloqueado.

Una forma más segura de comprobar esto sería mirar los eventos como los que se presentan en el siguiente ejemplo:

Ejemplo 2–1. Eventos generados periódicamente por el demonio Syslog de UNIX.

Jul 1 16:04:53 windbag -- MARK --

Jul 1 16:24:53 windbag -- MARK --

Jul 1 16:44:53 windbag -– MARK –-

Estos eventos son mensajes de estado generados periódicamente por el demonio Syslog de los sistemas UNIX. Esto nos asegura que el sistema está operando lo suficiente como para que el demonio Syslog escriba estos logs.

2.2.2 Detección de intrusos

Los dispositivos de seguridad de red como IPS, IDS o Firewalls generan logs que reflejan los eventos relacionados con la detección de intrusos, pero no solo estos dispositivos generan logs útiles para esta tarea.

Por ejemplo, el siguiente evento es generado por el servicio ssh de un equipo cualquiera conectado a la red:

Ejemplo 2–2. Evento generado por el servicio ssh.

Feb 23 11:41:37 myserver proxy: sshd[45983]: Failed password for ilegal user admin from 10.0.30.138 port 41063 ssh2ior.

Este evento muestra un intento fallido de acceso al sistema con el usuario “admin”. En el evento se indica que el usuario es ilegal debido a que no se corresponde con ningún usuario existente en el sistema. Esto podría deberse a un ataque usando un ssh-scanner, el cual intenta acceder al sistema usando un diccionario de pares usuario-contraseña que se usa comúnmente. Si además de este evento, encontráramos varios eventos de este tipo en un corto intervalo temporal, la evidencia del ataque es clara.

2.2.3 Análisis forense

El análisis forense es el proceso consistente en construir una imagen de lo que ha ocurrido una vez ha concluido un evento. Para construir esta imagen, es crítica la credibilidad de la información usada. Los logs pueden ser una parte esencial en el análisis forense.

Una vez guardados en la memoria, los logs no son alterados durante el transcurso del uso normal del sistema, por lo que pueden constituir una fuente de información más fiable que otros datos del sistema que son más susceptibles a ser alterados o corrompidos.

Además, normalmente los logs guardan una marca temporal que indica el momento en el que se generó. Gracias a esto, podemos crear una secuencia lógica de eventos ordenados en el tiempo.

Por si fuese poco, los logs presentan la ventaja de que pueden ser enviados a un servidor central de logs (mediante un servicio de gestión de logs). Esto nos proporciona una fuente de evidencia externa, y que puede suponer una fuente más fiable en caso de que se ponga en duda la integridad de la información de la fuente original (por ejemplo, si un intruso ha modificado o borrado los logs originales).

2.3 Formatos y sintaxis

La sintaxis y el formato de un archivo log define como se genera, transporta, almacena y analiza. En el caso más sencillo, cada evento en un log podría tratarse como una cadena de texto, y el interesado en consultar esos logs podría llevar a cabo una búsqueda de texto en los archivos de logs.

Tabla 2–1. Resumen de tipos de logs

	XML	Syslog	Fichero de texto	Binario
Modo de consumo	Procesamiento por ordenador.	Procesamiento manual en su mayoría.	Sólo procesamiento manual.	Sólo procesamiento por software.
Caso de uso	Seguridad.	Logging operacional y depuración.	Depuración.	Alto rendimiento.
Ejemplo	Cisco IPS	Mayoría de routers y switches.	Depurar aplicaciones.	Firewall Checkpoint.
Uso recomendado	Se usa cuando un conjunto de información estructurada necesita ser transferida a un consumidor para analizarla.	Se suele usar añadiéndole estructuras tipo clave-valor para simplificar el análisis automático.	Se suele usar añadiéndole estructuras tipo clave-valor para simplificar el análisis automático.	Suele usarse sólo en casos de necesidad de rendimiento muy alto.
Desventajas	Rendimiento bajo y mensajes log demasiado grandes.	La falta de estructura dificulta el análisis automático.	Es el caso más complejo y costoso para análisis automático.	No son legibles por humanos sin una herramienta para convertir los binarios en texto.

2.3.1 Syslog

Este formato de logs se le puede atribuir el tipo ASCII, ya que los logs están formados por este tipo de caracteres y se almacenan en ficheros que pueden abrirse con un simple editor de textos. A continuación, podemos ver un ejemplo del formato Syslog[2]:

[2] El término syslog se usa comúnmente para referirnos al protocolo de transporte de logs y al formato de logs.

Ejemplo 2–3. Evento en formato Syslog.

<165>1 2003-10-11T22:14:15.003Z mymachine.example.com eventslog - ID47 - An application event log entry...

A simple vista, podemos ver que el evento del ejemplo anterior nos indica el momento en el que fue generado (2003-10-11T22:14:15.003Z), el equipo (mymachine.example.com) y la aplicación (eventslog) que lo generó, y un mensaje ("An application event log entry..."). Como podemos observar, el log es fácilmente entendible por un humano, siempre que el mensaje sea claro.

2.3.2 IDMEF

Los servidores web, equipos de red, y la mayoría de las aplicaciones de todas las plataformas almacenan sus logs en ficheros de texto que pueden leerse fácilmente, no obstante, debemos hacer aquí una distinción entre formatos legibles por humanos y formatos basados en caracteres ASCII.

Por ejemplo, existen aplicaciones que generan sus eventos en formato XML. El formato XML, aunque está compuesto de caracteres legibles para los humanos, puede resultar difícil de entender cuando nos encontramos con numerosas etiquetas anidadas.

Ejemplo 2–4. Evento en formato IDMEF.

```
  <?xml version="1.0" encoding="UTF-8"?>
  <idmef:IDMEF-Message xmlns:idmef="http://iana.org/idmef"
version="1.0">
    <idmef:Alert messageid="abc123456789">
      <idmef:Analyzer analyzerid="bc-sensor01">
        <idmef:Node category="dns">
          <idmef:name>sensor.example.com</idmef:name>
        </idmef:Node>
      </idmef:Analyzer>
      <idmef:CreateTime ntpstamp="0xbc71f4f5.0xef449129">2000-03-
09T10:01:25.93464Z</idmef:CreateTime>
      <idmef:Source ident="a1a2" spoofed="yes">
        <idmef:Node ident="a1a2-1">
          <idmef:Address ident="a1a2-2" category="ipv4-addr">
            <idmef:address>192.0.2.200</idmef:address>
          </idmef:Address>
        </idmef:Node>
      </idmef:Source>
      <idmef:Target ident="b3b4">
        <idmef:Node>
          <idmef:Address ident="b3b4-1" category="ipv4-addr">
            <idmef:address>192.0.2.50</idmef:address>
          </idmef:Address>
        </idmef:Node>
      <idmef:Classification text="Ping-of-death detected">
        <idmef:Reference origin="cve">
          <idmef:name>CVE-1999-128</idmef:name>
```

```
          <idmef:url>http://www.cve.mitre.org/cgi-
bin/cvename.cgi?name=CVE-1999-128</idmef:url>
        </idmef:Reference>
      </idmef:Classification>
    </idmef:Alert>
  </idmef:IDMEF-Message>
```

Si comparamos el ejemplo anterior con el ejemplo 2-3, enseguida nos daremos cuenta de que el evento del ejemplo 2-4 presenta una dificultad mucho mayor para su comprensión por parte de un humano. En cambio, este último evento presenta la ventaja de facilitar el procesamiento por parte de un ordenador.

2.3.3 Windows Event Log

Además de formatos basados en texto, también existen formatos binarios. El ejemplo más común de este formato es "Windows Event Log", el cual para poder ser entendido necesita convertirse previamente a un formato legible por humanos. Esto se realiza mediante la herramienta "Visor de eventos" de Windows.

Aunque los logs basados en texto son más simples de entender para los humanos, existen razones por la que a veces se elige un formato binario: rendimiento y espacio.

Los log binarios necesitan menos capacidad de procesamiento para ser parseados[3], y tienen campos y tipos de datos claramente definidos, lo que permite un análisis más eficiente. En cambio, un parseador de logs basados en caracteres ASCII tiene que procesar más datos, y casi siempre se basan en coincidencia de patrones para extraer la información útil. Además, los logs binarios ocupan normalmente menos espacio en memoria (y pueden ser comprimidos), así que necesitan menos capacidad de procesamiento para darles formato y escribirlos en memoria.

2.3.4 Sintaxis

Todos los logs, sea cual sea su formato, tienen también una sintaxis. Cada log tiene una estructura propia, pero, basándonos en los formatos de logs más usados, podemos establecer un conjunto de campos que suelen aparecer en todos los eventos:

- Fecha y hora (timestamp) en la que se generó el mensaje. Esto nos permite ordenar los mensajes cronológicamente y analizar cómo se sucedieron los eventos.
- Equipo que generó el evento. Esto es muy útil cuando tenemos varios equipos enviando eventos a un servidor central, ya que nos permite distinguir a que equipo pertenece cada evento.
- Aplicación o componente que generó el evento. Nos permite distinguir dentro de cada equipo, que parte de este generó el mensaje.
- Rigor (severity), prioridad, o importancia del evento.
- Si el evento está relacionado con cualquier actividad que implique el uso de usuario, indicará el nombre de usuario usado para el acceso.

[3] Un "parser" en el lenguaje de la computación, es un analizador sintáctico.

3 CORRELACIÓN DE EVENTOS

We live in a society exquisitely dependent on science and technology, in which hardly anyone knows anything about science and technology.

- Carl Sagan-

El otro concepto del que hablaremos durante todo el trabajo, junto con el del capítulo anterior, es el de correlador de eventos.

3.1 Introducción a la correlación de eventos

La definición de evento en el diccionario es la siguiente: Acontecimiento, especialmente si es de cierta importancia. En el contexto de la informática, el término de evento también se utiliza para el mensaje que se genera cuando ha sucedido algo, estos mensajes son los logs. Ejemplos podrían ser un log generado por un analizador de tráfico, o el log que se genera debido al inicio fallido en un usuario.

El significado del término correlación se hace uso para las relaciones entre diferentes eventos. La correlación de eventos se realiza generalmente para obtener un conocimiento de la información de más alto nivel que nos puede proporcionar los eventos, por ejemplo, identificar situaciones extraordinarias, para identificar la causa raíz de un problema, o para realizar predicciones sobre el futuro y descubrir tendencias.

Existen diversos enfoques para la que puede ser útil la correlación de eventos, como análisis de datos de mercado, detección de fraude (por ejemplo, detectar los patrones de uso infrecuente de una tarjeta de crédito), análisis de logs del sistema (por ejemplo, agrupar mensajes similares y aumento de acontecimientos importantes) o análisis de gestión y fallas de red (por ejemplo, detectar la causa de un problema de red).

Aunque esta lista es corta, muestra que la correlación de eventos es un tema amplio, con numerosas aplicaciones. Pero en nuestro caso, discutimos la correlación de eventos principalmente desde la perspectiva del análisis de eventos en la red y registros del sistema. [2]

3.1.1 Operaciones

En esta sección, se presentarán las operaciones básicas que se utilizan para la correlación de eventos. Estas operaciones, pueden ayudar para la construcción de patrones más complejos para la correlación de eventos.

3.1.1.1 Compression

La compresión es la operación de reemplazar múltiples eventos idénticos por un solo evento. El evento por el que se sustituye debe contener el número de eventos por el cual se sustituyó.

De esta manera 1000 mensajes "ECHO reply" se convierte en una única alerta que sería "1000 ECHO reply han llegado"

3.1.1.2 Aggregation

La agregación es la operación de recolección de varios eventos en un solo evento. Por lo tanto, puede verse como una forma de compresión sin pérdida. A diferencia de la compresión, la agregación por lo tanto crea un nuevo evento, con un nuevo significado, que incluye los eventos agregados en lugar de reemplazarlos, como ocurre en la compresión).

3.1.1.3 Filtering

La filtración es la operación de eliminación de eventos con unas propiedades dadas en una secuencia de eventos.

3.1.1.4 Suppression

La supresión asocia prioridades con alarmas y permite que el sistema elimine una alarma de prioridad más baja si ha ocurrido un evento de mayor prioridad. Por tanto, sería más una propiedad de filtrado que de correlación.

3.1.1.5 Thresholding

El umbral es la operación de generar un evento si se supera un cierto umbral, o viceversa, si la tasa de evento cae por debajo de un umbral.

3.1.1.6 Escalation

La escalada es la operación de aumento de un parámetro de un evento (por ejemplo, prioridad) dadas unas condiciones.

3.1.1.7 Temporal relationship

Las operaciones de relación temporal, correlacionan eventos basados en el tiempo y en el orden en el que llegan. Por ejemplo, dos eventos llegan con una ventana de 5 minutos

3.1.1.8 Generalization

La operación de generalización se encarga de transmitir algo más general, en lugar del evento concreto. Aunque, no se genera ninguna nueva información, la generalización puede ser útil para detectar una causa común para varios eventos.

3.1.1.9 Specialization

La especialización es lo opuesto de la generalización. Por ejemplo, llega un evento de que un servicio, como un servidor web, no funciona. Entonces sería conveniente realizar reglas específicas para gestionar la falta de servicios.

3.1.1.10 Clustering

El clustering se define como una operación que genera un nuevo evento, combinando otras operaciones. Por ejemplo, agregar todos los eventos que llegaron en los últimos diez minutos si llega el evento A y además que exista el contexto B.

3.1.2 Tipos de eventos

Hay diferentes tipos de eventos, los cuales son:

- *Raw event:* es un evento, que se genera en su origen, es decir desde fuera del sistema de correlación.

- *Input event:* es un evento a la entrada del correlador. Por el contrario, un *output event* es un evento que se produce a la salida del correlador.
- *Compressed event:* es un evento que representa a varios eventos idénticos.
- *Derived event:* es un evento que es generado por el correlador, por ejemplo, porque ocurrió un determinado conjunto de eventos.
- *Timeout event:* es un evento generado durante un tiempo especificado. Características

3.1.3 Falsos positivos y falsos negativos

Un falso positivo es un evento que indica una situación extraordinaria, a pesar de no haber ningún problema, por ejemplo, un correo electrónico se clasifica como spam, aunque no es spam.

Un falso negativo es un evento que indica que la situación es normal, a pesar de que hay un problema, por ejemplo, un correo electrónico entra en la carpeta de entrada, a pesar de que es spam.

Si se comprueba un servicio y no se genera ningún evento, a pesar de que el servicio tiene un problema, esto puede también considerarse como un falso negativo. Normalmente, ningún mensaje se considera como si no tuviera ningún problema.

La solución a este último caso es simplemente introducir nuevos controles para cubrir el problema que se pasa por alto.

3.1.4 Real-time vs Offline

Los correladores de eventos pueden realizar su función en tiempo real con datos entrantes u offline, es decir, con datos ya guardados.

En el caso de los correladores de eventos para la monitorización de log, la función de en tiempo real es necesaria. Sin embargo, esto solo significa que los eventos son procesados en tiempo real, pero no es del todo necesario ya que en algunos casos es necesario retrasar los eventos un corto tiempo. Dependiendo de si los eventos pueden ser modificados o no.

La función offline es útil en el caso de que fuera necesario encontrar patrones de eventos en ficheros con una gran cantidad de datos.

3.1.5 Centralizado vs Distribuido

Los eventos que se van generando, normalmente, se producen en fuentes que pueden estar distribuidas en una red, lo que nos da a una arquitectura distribuida. La ventaja es el rendimiento y la escalabilidad, y la facilidad de acceder a información adicional de las fuentes.

En el caso de una arquitectura centralizada tenemos la ventaja de que es más fácil su gestión.

3.1.6 Contextos

Los contextos son condiciones interrelacionadas en las que existe algo o se produce. En otras palabras, un contexto existe u ocurre cuando es creado por una regla y en el que actúan como un almacén de eventos.

Por ejemplo: podemos crear un contexto en una regla para recordar de que ha ocurrido algo y luego utilizarlo en otra para saber si ha ocurrido algo anteriormente. Esto es, si llega un paquete con una IP se guarda en un contexto y luego si llega otro con la misma IP se comprueba el contexto, por si había llegado anteriormente para crear un aviso o alerta de que posiblemente se esté haciendo un escaneo de puertos o similar.

3.2 Expresiones regulares

Cuando manejamos texto en algún tipo de lenguaje, una de las operaciones más comunes es la búsqueda de una subcadena. Si la cadena que buscamos es fija, por ejemplo, en Python los métodos find(), index() o similares nos ayudarán. Pero si buscamos una subcadena con cierta forma, este proceso se vuelve más

complejo.

Al buscar direcciones de correo electrónico, números de teléfono, validar campos de entrada, o una letra mayúscula seguida de dos minúsculas y de 5 dígitos entre 1 y 3; es necesario recurrir a las Expresiones Regulares, también conocidas como patrones. Las cuales son necesarias en casi todos los correladores para la captura de eventos. [3]

3.2.1 Patrones

Las expresiones regulares son un potente lenguaje de descripción de texto, que nos ayudará a parsear los diferentes registros de log. Y no existe un lenguaje moderno que no permita usarlas.

Utilizándolas podemos buscar una subcadena al principio, en medio o al final del texto. Permite, además, capturar aquellos trozos del texto que coincidan con la expresión para guardarlos en una variable o reemplazarlos por una cadena predeterminada.

Estos son algunos aspectos básicos de las expresiones regulares:

3.2.1.1 Clases de caracteres

Se utilizan cuando se quiere buscar un carácter dentro de varias opciones. Una clase se delimita entre corchetes y lista posibles opciones para el carácter que representa:

- [abc]: coincide con a, b, o c
- [387ab]: coincide con 3, 8, a o b
- niñ[oa]s: coincide con niños o niñas.

Debemos escaparlas colocándoles una contrabarra (\) delante para buscarlos explícitamente

Para evitar errores, en caso de que queramos crear una clase de caracteres que contenga un corchete, debemos escribir una contra barra \ delante, para que el motor de expresiones regulares lo considere un carácter normal: la clase [ab\[] coincide con a, b y [.

3.2.1.2 Rangos

Si queremos encontrar un número, podemos usar una clase como [0123456789], o podemos utilizar un rango. Un rango es una clase de caracteres abreviada que se crea escribiendo el primer carácter del rango, un guion y el último carácter del rango.

3.2.1.2.1 Ejemplo

Algún ejemplo de rangos:

- [a-c]: equivale a [abc]
- [0-9]: equivale a [0123456789]
- [a-d5-8]: equivale a [abcd5678]

3.2.1.3 Rango negado

Así como podemos listar los caracteres posibles en cierta posición de la cadena, también podemos listar caracteres que no deben aparecer. Para lograrlo, debemos negar la clase, colocando un circunflejo después del corchete izquierdo: [^abc] coincide con cualquier carácter distinto a a, b y c.

3.2.1.4 Clases predefinidas

Existen algunas clases que se usan frecuentemente y por eso existen formas abreviadas para ellas. En Python, así como en otros lenguajes, se soportan las clases predefinidas de Perl. Algunos ejemplos son:

- \d: equivale a [0-9]

- \s: caracteres de espacio en blanco (espacio, tabulador, etc.)
- \w: letras minúsculas, mayúsculas, números y guion bajo (_)
- Además, existe una clase de que coincide con cualquier otro carácter. Ya sea letra, número, o un carácter especial. Esta clase es el punto: ".": coincide con cualquier carácter.

3.2.1.5 Cuantificadores

Son caracteres que multiplican el patrón que les precede. Mientras que con las clases de caracteres podemos buscar un dígito, o una letra; con los cuantificadores podemos buscar cero o más caracteres. Los cuantificadores son:

- ?: coincide con cero o una ocurrencia del patrón. Dicho de otra forma, hace que el patrón sea opcional
- +: coincide con una o más ocurrencias del patrón
- *: coincide con cero o más ocurrencias del patrón.
- {x}: coincide con exactamente x ocurrencias del patrón
- {x, y}: coincide con al menos x y no más de y ocurrencias. Si se omite x, el mínimo es cero, y si se omite y, no hay máximo.

3.2.1.5.1 Ejemplo

- [a-z]{3,6}: entre 3 y 6 letras minúsculas.
- \d{4,}: una cadena que tenga al menos 4 dígitos.
- .* : cualquier cadena, de cualquier longitud.

3.3 Arquitectura general

La arquitectura general que nos podemos encontrar en los correladores de eventos es la siguiente:

Figura 3-1. Arquitectura de un correlador

Algunos detalles de este esquema:

- Una fuente o host, que puede ser un ordenador, router…, envía su información o eventos al servidor Syslog o a un fichero de texto.
- Se realiza una recolección de los log en un formato determinado (XML, ficheros de texto…).
- Se normalizan para la plataforma o sistema de correlación.
- Se correlan los eventos recibidos y se toma una serie de decisiones:

  - **Mensaje descartado**. Los mensajes que son solo informativos, se descartan y no se almacenan en la base de datos. En dicha base de datos se almacenan los log y los eventos de los que se quieren informar.
  - **Mensaje almacenado**. Los mensajes que se consideran importantes se almacenan en la base de datos de log.
  - **Alarma**. Si hay un evento que genera alguna alarma, se guarda en la base de datos de alarmas. Además de esto, pueden generarse acciones adicionales, tales como enviar un email o mostrar un mensaje en la consola de monitoreo.
- Y se monitorizan por un usuario o sistema final.

Con todo esto podemos tener una visión general para introducir a continuación los sistemas elegidos.

4 DESCRIPCIÓN DEL SOFTWARE ELEGIDO

Computer Science is embarrassed by the computer.

- Alan Perlis-

Actualmente podemos encontrar un amplio abanico de sistemas de correlación de eventos. Para este proyecto nos vamos a centrar en tres: SEC, Nxlog y Prelude.

En cada apartado hablaremos de sus características y lo que nos puede ofrecer. En la siguiente tabla podemos ver una visión general:

Tabla 4–1. Características generales de los tres sistemas de correlación

Software	Dominio	Lenguaje	Técnica de correlación
SEC	Carácter general	Perl	Reglas basadas en texto plano
Nxlog	Monitorización de log	C	Reglas basadas en texto plano
Prelude	SIEM	C, Python	Reglas basadas en Python

4.1 SEC

Simple Event Correlator es una herramienta de correlación de eventos de código abierto escrito en Perl, que utiliza un enfoque basado en reglas para el procesamiento de eventos, permitiendo reconocer los eventos de entrada mediante el uso de expresiones regulares o subrutinas de Perl. Se ejecuta como un proceso único, no requiere ningún entorno gráfico, y tiene un consumo moderado de la memoria de la CPU. Funciona en cualquier plataforma UNIX con la distribución estándar de Perl, sin dependencia de cualquier otro software. También se ha utilizado en los sistemas Windows, pero requiere la adición de ActivePerl o CygWin Perl.

SEC toma los eventos desde diferentes fuentes, como:

- Las líneas de un fichero.
- O mediante la entrada estándar del terminal.

SEC puede producir una salida mediante la ejecución de programas externos (por ejemplo, snmptrap o correo), escribiendo en los archivos indicados, mediante el envío de datos a través de TCP y UDP, llamando a subrutinas de Perl, etc.

Además, también permite la creación de eventos y contextos, junto con las operaciones de correlación básicas proporcionadas por SEC, permite la detección de eventos. Aunque las operaciones individuales son bastante

simples, la posibilidad de combinar con otras reglas, contextos, expresiones regulares y Perl scripts externos, permite una correlación bastante compleja y hace a SEC muy versátil. A pesar de su simplicidad, SEC tiene una base de usuarios y es ampliamente utilizado con éxito incluso en grandes redes. [4]

4.1.1 Operaciones de correlación

Las operaciones de correlación de SEC son:

- **Single**: esta regla en el momento que la regla captura un *input event* y es evaluado el contexto, si es que existe, se ejecuta una acción.
- **SingleWithScript**: igual que *Single*, pero añadiendo la ejecución de un script.
- **SingleWithSuppress**: igual que *Single*, pero durante un tiempo t determinado se ignoran los siguientes eventos.
- **Pair:** igual que *Single*, pero se tienen que cumplir dos condiciones. Se ignoran los eventos coincidentes hasta que llegue el siguiente *inpunt event*.
- **PairWithWindow**: igual que el anterior caso, pero se tienen que cumplir dos condiciones y en un tiempo como máximo.
- **SingleWithThreshold**: igual que *Single*, pero se tienen que cumplir una condición tantas veces como indique el campo thresh antes de que se pueda ejecutar la acción.
- **SingleWith2Thresholds**: en este caso, cuenta todas las capturas que hace de los *input* events durante t1 segundos y si se supera un determinado umbral, ejecuta una acción. Luego se inicia el conteo de eventos otra vez y si su número en t2 segundos cae por debajo del umbral de la segunda, ejecuta otra acción.
- **Suppress**: suprime las coincidencias de los *input* events, una vez que se ha capturado un evento, durante un tiempo t.
- **Calendar**: ejecuta una acción en un tiempo específico.

4.1.2 Reglas

Suponemos que SEC se inicializa con el siguiente comando:

```
/usr/bin/sec --conf=/etc/sec/sshd.rules --input=/var/log/secure
```

Con el cual estamos indicando que la entrada que se va a monitorizar es /var/log/secure y las reglas con las que se va a correlar un fallo de logeo se encuentran en /etc/sec/sshd:

Ejemplo 4–1. Ejemplo de regla en SEC

```
type=SingleWithThreshold
ptype=RegExp
pattern=sshd\[\d+\]: Failed .+ for (\S+) from [\d.]+ port \d+ ssh2
desc=Tres fallos de logeo sobre SSH en menos de un minuto para el usuario $1
action=pipe '%s' /bin/mail -s 'SSH login alert' root@localhost
window=60
```

thresh=3

El campo pattern de la regla define el patrón a reconocer en la entrada mientras el campo ptype define el tipo de patrón el cual es una expresión regular. Suponemos que el usuario Luis introduce mal sus datos, nos aparecería el siguiente mensaje en /var/log/secure:

Ejemplo 4–2. Mensaje de log por fallo de inicio de sesión

Dec 16 16:24:59 myserver sshd[13685]: Failed password for luis from 10.12.2.5 port 41063 ssh2ior.

Si esto ocurriera 3 veces en menos de un minuto, tal y como indica el campo thresh, en menos de 60 segundos, mandaría un correo al root para informar de este problema.

Se amplía más información en el *apartado 4* y en el *Anexo A*.

4.2 Nxlog

Nxlog es una solución de gestión multiplataforma de log de alto rendimiento orientada a la solución de estas tareas y hacerlo todo en un solo lugar.

Nxlog puede trabajar en un entorno heterogéneo de recogida eventos de log entre miles de diferentes fuentes, en muchos formatos. Nxlog puede aceptar los registros de eventos de TCP, UDP, archivos, bases de datos y otras fuentes en diferentes formatos como Syslog. Se puede realizar la reescritura de logs, correlación, alertando, búsqueda de patrones, ejecutar los trabajos programados, o incluso rotación de registros.

Su arquitectura multi-hilo permite la entrada, registro de tareas de procesamiento y de salida para ser ejecutadas en paralelo. El uso de una capa de alto rendimiento de E/S es capaz de manejar miles de conexiones de cliente simultáneas. Además, puede procesar las fuentes de entrada en un orden de prioridad. esto puede ayudar a evitar la pérdida de mensajes UDP. En caso de congestión de la red u otros problemas de transmisión de logs, Nxlog puede guardar los mensajes en el disco o en la memoria usando módulos. Es posible extender aún más sus funcionalidades mediante el uso de módulos cargables, de manera similar al de un servidor Web Apache. Además del modo de procesamiento de logs en tiempo real, se puede utilizar para procesar logs en modo offline. El lenguaje de configuración sigue un estilo similar al de Apache.

Por otro lado, Nxlog presta bastante atención a la seguridad ya que puede ejecutarse sin modo superusuario, y además proporciona un transporte TLS/SSL por lo que los mensajes estarán seguros en el transporte.

Además, el módulo de correlación de Nxlog está inspirado en SEC y podemos encontrar una referencia en el manual de Nxlog [5] en la que se dice:

- SEC utiliza expresiones regulares que pueden convertirlo en un sistema bastante lento en el caso de que haya demasiadas reglas de correlación. Por el contrario, Nxlog puede correlacionar mensajes previamente con campos. Por ejemplo, desde el comparador de patrón y el analizador Syslog sin requerir el uso de las expresiones regulares. Por lo tanto, el testeo de condiciones puede ser mucho más rápido cuando se utiliza una simple comparación en lugar de la coincidencia de patrones basado en expresiones regulares.
- Pero la más importante es que el modulo de correlación de Nxlog está escrito en C y SEC en Perl lo que puede dar un mayor rendimiento

4.2.1 Operaciones de correlación

Las operaciones de correlación en Nxlog están consideradas como módulos:

- **Simple**: este módulo, ejecuta simplemente una acción sin ningún tipo de condición. La condición puede estar dentro de la propia ejecución.
- **Suppressed**: suprime las coincidencias de los *input events* una vez que se ha capturado otro evento durante un tiempo t.
- **Pair**: esta regla/módulo, evalúa un *input event* si coincide con la regla espera un tiempo t en el cual se tiene que evaluar otra condición. En el caso que se encuentre la segunda coincidencia se ejecuta una acción.
- **Absence**: en este caso, ocurre lo contrario que en la regla *Pair*, si la segunda coincidencia no se encuentra en un tiempo t, se ejecutaría una acción
- **Thresholded**: este módulo, ejecuta una acción en el caso de que se supere un cierto umbral en un tiempo t.
- **Stop**: este módulo, para la evaluación de condiciones durante un tiempo t si se cumple una cierta condición.

4.2.2 Reglas

La configuración de las reglas se basa en módulos. Un ejemplo de ella:

Ejemplo 4–3. Archivo de configuración de Nxlog

```
<Input in>
    Module      im_file
    File        "modules/processor/evcorr/testinput_evcorr2.txt"
    SavePos     FALSE
    ReadFromLast FALSE

        Exec    if ($raw_event =~ /^(\d\d\d\d-\d\d-\d\d \d\d:\d\d:\d\d) (.+)/)
{  \
                    $Message = $2;                          \
                    $raw_event = $Message;                  \
                }
</Input>

<Output out>
    Module      om_file
    File        'tmp/output'
</Output>

<Processor evcorr>
    Module        pm_evcorr

    <Pair>
        TriggerCondition  $Message =~ /^Primera-condicion/
        RequiredCondition $Message =~ /^Segunda-condicion/
        Interval   30
        Exec       $raw_event = "Se ha capturado";
    </Pair>

</Processor>

<Route 1>
    Path        in => evcorr => out
```

```
</Route>
```

A continuación, se explica la utilidad de los módulos que principalmente vamos a utilizar:

- El módulo *Input* se utiliza para indicar la fuente de donde se va a recoger los datos.
- El módulo *Output* se utiliza para indicar hacia donde quiere que se redirija la salida.
- El módulo *Pm_evcorr* es el que principalmente vamos a utilizar, ya que es el que nos va a facilitar la correlación de eventos.
- El módulo *Route* se utiliza para definir el flujo y el orden de procesamiento de log, además de indicar la prioridad con la que se quiere realizar.

Se amplía más información en el *apartado 4* y en el *Anexo B*.

4.3 Prelude

Prelude Correlator es un motor de correlación basado en reglas de Python. Tiene la capacidad para conectar y buscar alertas desde un servidor remoto llamado Prelude-Manager y correlacionar alertas entrantes basados en las reglas proporcionada.

Inicialmente, el lenguaje de Prelude Correlator fue inspirado por SEC, evolucionando para utilizar un lenguaje de programación real. En este punto, se decide cambiar a un motor de reglas basado en Python, que proporciona la flexibilidad necesaria para escribir reglas de correlación.

Prelude es un sistema universal SIEM el cual recopila, normaliza, ordena, agrega, correla y notifica todos los eventos relacionados con la seguridad a través de una interfaz gráfica llamada Prewikka. Aunque, esta salida puede ser generada en otros formatos.

Prelude no proporciona ningún agente, pero es compatible con numerosos programas como OSSEC, Short, PAM… A la vez también se puede encargar de los logs de distintos dispositivos con el plugin LML que proporciona Prelude. Los mensajes que se han recolectado son normalizados y estandarizados en el formato IDMEF que posteriormente serán correlados mediante Prelude Correlator.

4.3.1 Arquitectura

La arquitectura que presenta Prelude es un poco más compleja que los dos sistemas de correlación anteriores, por eso vamos a mostrar la arquitectura concreta de este:

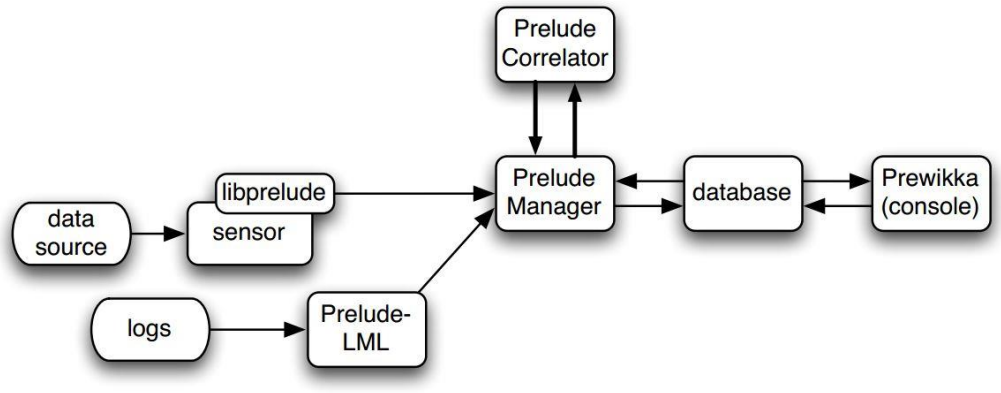


Figura 4-1. Arquitectura de Prelude

Algunos aspectos que comentar sobre esta arquitectura:

- Los datos proceden de:
  - Sensores como: Snort, Suricata, entre otros.
  - O del análisis de logs a través de Prelude LML.
- Estos datos llegan estandarizados al manager, mediante el estándar IDMEF.
- Se guardan en una base de datos para:
  - Correlarlos, mediante Prelude Correlator.
  - Mostrarlos por pantalla mediante la interfaz Prewikka.

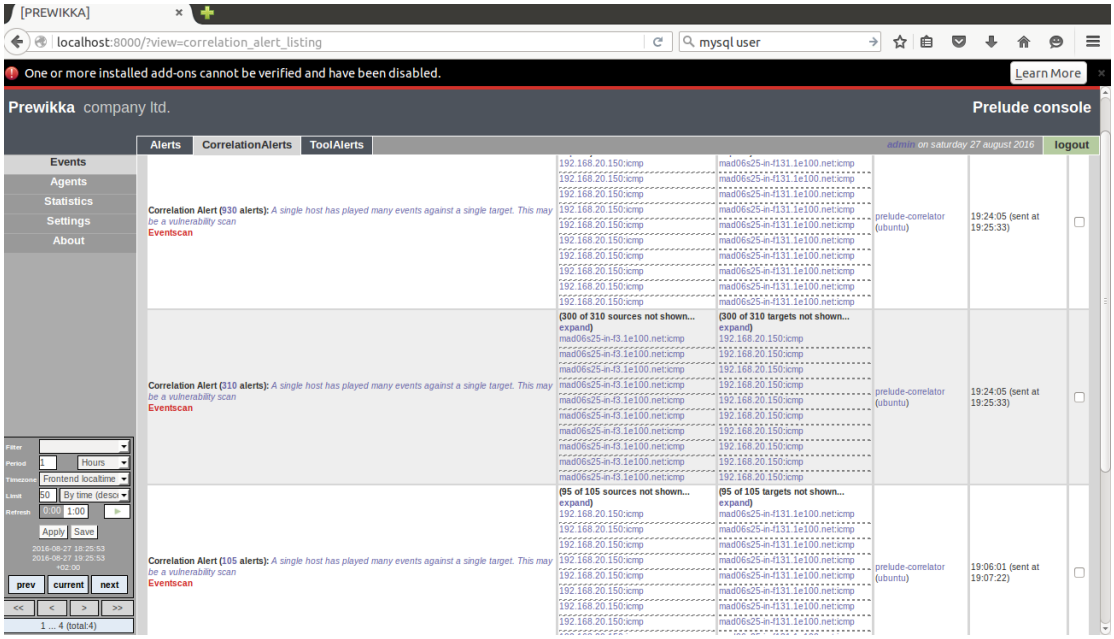


Figura 4-2. Interfaz Prewikka

4.3.2 Reglas

Las reglas en Prelude se consideran como plugins, son bastantes más difíciles (en comparación con los dos sistemas anteriores) de escribir ya que hay que tener algunos conocimientos en Python.

Ejemplo 4–4. Regla que alerta los sábados, domingos y todos los días de 18-9h

```
import time
from PreludeCorrelator.pluginmanager import Plugin


class BusinessHourPlugin(Plugin):
    def run(self, idmef):

        t = time.localtime(int(idmef.Get("alert.create_time")))
```

```
          if not (t.tm_wday == 5 or t.tm_wday == 6 or t.tm_hour < 9 or
t.tm_hour > 17):
                    return

          if idmef.Get("alert.assessment.impact.completion") !=
"succeeded":
                    return

          ca = IDMEF()
          ca.addAlertReference(idmef)
          ca.Set("alert.classification",
idmef.Get("alert.classification"))
          ca.Set("alert.correlation_alert.name", "Critical system
activity on day off")
          ca.alert()
```

Algunos aspectos a comentar:

- Cada plugin consiste en una clase que recibe el evento IDMEF a través de Prelude Correlator.
- El método set permite establecer, en una ruta determinada, un valor dentro de un objeto IDMEF.
- El método alert permite enviar al manager un objeto, lo cual es útil para generar alertas de correlación.

Además de estos métodos, existen otros como el método match (para realizar comparaciones) o los contextos como ya hemos visto en otros sistemas de correlación.

4.3.2.1 Contextos en Prelude Correlator

En este apartado se va a hablar sobre los contextos en Prelude Correlator, ya que pueden ser más difícil de comprender que en los otros sistemas.

Un contexto es una variable que guarda "la traza" del estado de un plugin. Existe una clase en Python llamada *Context*, que facilita la escritura de reglas de correlación.

Ejemplo 4–5. Contexto en Prelude Correlator

```
  ctx = Context(("SCAN_EVENTSTORM", saddr), { "expire": 120,
"threshold": 150, "alert_on_expire": True }, update = True, idmef =
idmef)
```

Está formado por:

- Un mensaje IDMEF, que estará embebido y será usado para la correlación.
- Un nombre, que identifica al contexto.

En el ejemplo 3-5, el campo: ("SCAN_EVENTSTORM", saddr) nos indica que el contexto se llamará

SCAN_EVENTSTORM_(el valor de saddr). Si saddr vale x.x.x.x, el contexto se llamaría SCAN_EVENTSTORM_x.x.x.x.

- Un campo *expire* opcional con un temporizador, el contexto se destruye si expira dicho temporizador.
- Y otro campo *threshold* opcional con un umbral, que será comprobado con el método *getUpdateCount.*
- El campo *alert_on_expire* nos indica que si un contexto expira se enviará una alerta IDMEF.
- El campo *update* nos indica que se reseteará el contexto en caso de que haya expirado el temporizador.

Se amplía más información en el *apartado 4* y en el *Anexo C*.

5 DESCRIPCIÓN DEL EXPERIMENTO

I never took a computer science course in college, because then it was a thing you just learned on your own.

- Mitchel Resnick -

Una vez terminada la fase de investigación de este proyecto (gran parte de esta se realizó en paralelo a la fase de desarrollo), comenzamos con este capítulo la fase de desarrollo.

El experimento que vamos a llevar a cabo se realizará en los tres sistemas, mediante un script. La razón de utilizar dicho script es, la de llevar a cabo la medición de los tiempos que dura en capturar el correlador el evento y alertar, para comprobar el rendimiento que ofrece cada sistema.

5.1 Descripción

El escenario que se va desarrollar, consiste en la detección de una petición masiva desde un único dispositivo a un host en concreto. Este escenario también se podría ampliar para otros casos. Por ejemplo, para la petición masiva desde diferentes hosts a uno en concreto o, desde otro punto de perspectiva, detectar cuando se inicia sesión de un usuario erróneamente varias veces. Incluso la combinación de las anteriores.

En nuestro caso, como queremos comprobar el rendimiento y para facilitar el proceso, se va a realizar la correlación a una respuesta masiva de un ping desde un host específico.

La entrada en SEC y Nxlog será la de un fichero, que contiene el análisis en tiempo real del tráfico de la red, gracias a la ayuda de un sniffer de paquetes como puede ser Snort o Tcpdump. En el caso de Prelude, también se hace uso de ellos, pero el correlador lee directamente los eventos de la base de datos que le proporciona el manager.

A continuación, se procederá a explicar cómo se ha realizado el script, y las reglas de correlación en cada uno de los sistemas.

5.2 Script

En este script, lo que se lleva a cabo es la medición de los tiempos que dura en capturar el correlador el evento y alertar. El script, debe ser ejecutado en modo superusuario ya que se hace uso de un ping masivo, es el siguiente:

```
#!/bin/bash
cd /home/antonio/TFG

ping -f google.es -c 200

inicio_ns=`date +%s%N`
inicio=`date +%s`

until grep -q "REPLY" salida; do
  echo "Nada encontrado por ahora"
done

fin_ns=`date +%s%N`
fin=`date +%s`
let total_ns=$fin_ns-$inicio_ns
let total=$fin-$inicio
echo "Ha tardado: -$total_ns- nanosegudos, -$total- segundo/s"
```

Elementos a comentar:

- La opción -f en el comando ping indica que se realiza un ping masivo
- La opción -c indica cuantos pings se quiere enviar. En nuestro caso 200, ya que los correladores deberán detectar cuando se realice la respuesta de los 150 primeros.
- Una vez realizado dicho ping, se inicia el contador. Para comprobar en el fichero de salida del correlador, en este caso llamado "salida", cuando es detectado el evento que estábamos buscando. Mientras no se haya encontrado nada, aparecerá por pantalla un mensaje de que no se ha encontrado.
- En el momento en el que el correlador muestre el evento, este lo detectará y se parará el contador.
- Por último, se mostrará por pantalla los tiempos. Se consideran dos:
    - *total_ns*: el tiempo en nanosegundos.
    - *total*: el tiempo segundos

Esto último se realiza de esta manera ya que hay correladores que pueden tardar menos de un segundo y otros que pueden llegar hasta los 100 segundos.

El motivo de haber seleccionado un ping de este tipo es para poner a prueba el funcionamiento de las ventanas en los correladores, nosotros estableceremos la ventana en 8 segundos. Ya que, este ping, puede realizar 200 peticiones en menos de 2 segundos, dependiendo de la red y del dispositivo el cual se esté realizando.

5.3 SEC

La regla utilizada para llevar a cabo este experimento en SEC es:

```
#
#Ataque ICMP desde una misma IP
#

type=SingleWithThreshold
ptype=RegExp3
```

```
 pattern=^\d{2}\/\d{2}-\d{2}:\d{2}:\d{2}\.\d+
((?:\d{1,3}\.){3}\d{1,3}) ->
(?:\d{1,3}\.){3}\d{1,3}.*?\nICMP.*?\nType:0\s+Code:0\s+ID:\d+\s
+Seq:\d+\s+ECHO REPLY
 desc=150 ECHO REPLY han llegado desde la IP: $1
 action=write - %s
 thresh=150
 window=8
```

Para la situación que queremos detectar hemos tenido las siguientes consideraciones:

- El campo *type indica* el tipo de regla que se va a usar, en nuestro caso: *SingleWithThreshold*. Ya que nos facilitará mucho la detección.
- En el campo *ptypte* se debe indicar el tipo de expresión regula que se va a utilizar. Normalmente se hace uso de RegExp, pero en nuestro caso al tener el fichero de entrada tres líneas por cada evento, nos llevan a utilizar el tipo RegExp3
- El campo *pattern* indica el patrón con el que se quiere capturar el evento, en el cual se hace uso de las expresiones regulares vistas en el apartado 2.3. Además, se puede capturar aquellos trozos del texto, rodeados entre paréntesis, que coincidan con la expresión para guardarlos en una variable y utilizarlo luego en la descripción o dentro de un contexto.
- El campo *desc* nos proporciona una descripción del evento capturado. En nuestro caso la captura de 150 ECHO reply desde un host especifico.
- El campo *action* es la acción que aplicamos, en nuestro caso es la de escribir a la salida del correlador la descripción del evento. Se podría realizar el envío de un correo para que le sea notificado a un supervisor.
- El campo *thresh* indica cuantos eventos del patrón indicado se tienen que dar para que se ejecute la acción.
- El campo *window* indica la ventana en la que se tiene que producir el evento. En nuestro caso una ventana de 8 segundos.

Por otro lado, la entrada y salida del correlador se indica mediante parámetros a la hora de su ejecución.

5.4 Nxlog

La regla utilizada para llevar a cabo este experimento en Nxlog es:

```
<Extension fileop>
    Module xm_fileop
</Extension>

<Input in4>
    Module im_file
    File "/home/antonio/TFG/entrada"
    SavePos FALSE
    ReadFromLast TRUE
```

```
      Exec if ($raw_event =~ /^\d\d:\d\d:\d\d.(.+)/) { \
                      $Message = $1;    \
                      $raw_event = $Message;       \
                   }
       exec if $Message =~ /IP (\S{1,}) > \S{1,}:/ $IP=$1;
  </Input>

  <Output out4>
      Module om_file
      File "/home/antonio/TFG/salida"
  </Output>

  <Processor evcorr>
      Module   pm_evcorr

      <Thresholded>
         Condition  $Message =~ /ICMP echo reply/
         Threshold  150
         Interval   8
         Context $IP
         Exec file_write("/home/antonio/TFG/salida2", "150 ECHO
REPLY han llegado desde la" +$IP +"\n");
      </Thresholded>

  </Processor>

  <Route 4>
  Path in4 => evcorr => out4
  </Route>
```

Como ya hemos comentado en el apartado 3.2 Nxlog está formado por módulos y para la situación que queremos detectar hemos tenido las siguientes consideraciones:

- La extensión *fileop* nos facilita la escritura externa en un archivo con la función file_write() utilizada en el módulo pm_evcorr.
- En el módulo *input* indicamos:
  - La fuente de entrada al correlador.
  - El campo *SavePos* indica que Nxlog deberá guardar la posición de lectura del fichero en el caso de que se salga de Nxlog.
  - El campo *ReadFromLast* indica que Nxlog solamente leerá los eventos que hayan sido recibidos después de su iniciación.
  - Indicamos que queremos guardar en la variable *Message* cada línea de un *input event* que empiece por `\d\d:\d\d:\d\d..`
  - Además, se guarda el valor de la IP con que llega el mensaje para utilizarla en el contexto.
- En el módulo *output* indicamos el fichero de salida.
- En el módulo *pm_evcorr* indicamos las condiciones de correlación. Las cuales son las mismas que en el anterior caso.
- Y, por último, el módulo *route* nos indica el flujo y el orden de procesamiento del log.

5.5 Prelude

La regla o plugin utilizado para llevar a cabo este experimento en Prelude Correlator es:

```
  from PreludeCorrelator.context import Context
  from PreludeCorrelator.pluginmanager import Plugin

  class PluginEventoEscaneo(Plugin):
      def run(self, idmef):
          fuente =
idmef.Get("alert.source(*).node.address(*).address")
          destino =
idmef.Get("alert.target(*).node.address(*).address")

          if not fuente or not destino:
              return

          for saddr in fuente:
              for daddr in destino:
                  ctx = Context(("Evento_Escaneo", saddr,
daddr), { "expire": 8, "threshold": 150, "alert_on_expire":
True }, update = True, idmef=idmef)
                  if ctx.getUpdateCount() == 0:
                      ctx.Set("alert.correlation_alert.name",
"150 paquetes han llegado")
                      ctx.Set("alert.classification.text",
"EventoDeEscaneo")

                      ctx.Set("alert.assessment.impact.severity
                      ", "high")
```

Elementos a comentar sobre el anterior plugin:

- Se recibe a través de la clase *PluginEventoEscaneo* un mensaje IDMEF de un sensor, en este caso Snort.
- La función run() toma un único argumento: el mensaje IDMEF recibido por Prelude Correlator.
- Inicialmente empezamos por recuperar una lista de todas las direcciones de las fuentes y destinos dentro de la alerta recibida guardándolas en las variables fuente y destino.
- Se comienza a recorrer la lista de fuentes y se ira creando, o recuperando (si ya existe), el contexto asociado a cada dirección de origen. El contexto caducara a los 8 segundos, y el contador interno de umbral se establece en 150, como en los anteriores casos. Si se recibe un mensaje IDMEF con la dirección de origen "x.x.x.x", el contexto creado se llamará Evento_Escaneo_x.x.x.x.
- Después de crear/actualizar el contexto, comprobamos si el contexto ha sido creado (y no actualizado). Una vez llegado el contador del umbral a 0, inicializaremos a parte la alerta con el método set.

6 RESULTADOS DEL EXPERIMENTO Y DISCUSIÓN

If data had mass, the Earth would be a black hole.
- Stephen Marsland -

Después de haber desarrollado en el apartado anterior el experimento, en este capítulo mostraremos los resultados obtenidos, además de una discusión de los mismos y de los sistemas utilizados.

Hay que tener en cuenta que los experimentos se han desarrollado sobre un ordenador personal, cuando estos sistemas normalmente suelen funcionar sobre servidores con un gran potencial. Además de haber estado corriendo el sistema operativo sobre una máquina virtual, lo cual, no es lo mismo que esté funcionando sobre un sistema operativo real.

6.1 Resultados

Los resultados obtenidos del experimento desarrollado en el apartado anterior se mostrarán tanto en segundos como nanosegundos, ya que hay un sistema correlador que se demora en alertar el evento.

Tabla 6–1. Tiempos de respuesta obtenidos de los experimentos de la sección 4

Software	Tiempo en segundos	Tiempo en nanosegundos
SEC	0.5	549697408
Nxlog	0.4	352369035
Prelude Correlator	128	127937785664

6.2 Discusión

Este apartado se dividirá en dos subapartados: uno en el que se tratará los datos obtenidos en el apartado anterior y otro apartado en el que se discutirá los beneficios y deficiencias que pueden tener cada uno.

6.2.1 Resultados

Los datos obtenidos son muy interesantes, sobre todo en el caso de Prelude.

6.2.1.1 SEC y Nxlog

Los resultados obtenidos entre SEC y Nxlog son muy parecidos, recordando lo que se nos indicaba en el manual:

- El modulo de correlación de Nxlog está escrito en C, y SEC en Perl, lo que puede dar un mayor rendimiento. [5]

Se puede observar un menor tiempo, pero no muy excesivo. Si llevamos este caso al análisis con cientos de reglas y distintos logs, puede ser que esa décima de diferencia sea bastante significativa en un sistema crítico.

6.2.1.2 Prelude

El resultado que nos arroja Prelude, nos puede resultar demasiado extraño. Ya que, a pesar de que la ventana es menor de 10 segundos para capturar el evento, este la supera con creces. Pero se debe a un motivo, y es que este sistema tiene una versión comercial.

En nuestro caso estamos haciendo uso de la versión Open Source, destinada a la educación o test de funcionamiento. Lo que conlleva a sus desarrolladores a limitar este producto en el sentido de que pausa la salida durante un tiempo, aunque se haya detectado el evento.

Lo cual nos lleva a no usar Prelude y si SEC o Nxlog, si queremos destinar uno de estos sistemas a la producción de entornos críticos o de importantes dimensiones.

6.2.2 Sistemas

Una vez finalizado todo el trabajo de investigación y desarrollo se discutirá los beneficios y deficiencias que pueden tener cada software.

6.2.2.1 SEC

En el caso de SEC hemos visto que es un sistema de carácter general. Está diseñado tanto para pequeñas como medianas empresas. SEC se mantiene entre las herramientas de correlación de eventos de gama baja y gama alta, es decir, sistemas comerciales como Cisco MARS.

Aunque, no es tan potente como las herramientas de gama alta, SEC cumple con 70-80% de los requisitos de correlación de eventos de los entornos no empresariales. E incluso podría ser adecuado para algunas organizaciones que no necesitan las poderosas capacidades de las herramientas de gama alta.

6.2.2.1.1 Ventajas

Algunas ventajas son:

- La configuración de las reglas es muy elemental y de fácil aprendizaje.
- La instalación que conlleva es la más básica entre los tres softwares, ya que no piden muchos requisitos, es: descargar, configurar reglas e iniciar.
- El soporte, en lo que se refiere a ayudas de configuración, es muy competente. Ya que tiene una *mailing list*[4] gestionada directamente por el creador del propio software, Risto Vaarandi.

6.2.2.1.2 Inconvenientes

Algunos de los inconvenientes que hemos observado son:

- Puede ser tedioso cuando quieres hacer alguna regla más allá de una comparación o un simple filtrado. Ya que tienes que hacer uso de scripts en Python.
- Solo proporciona los paquetes, para su instalación, para las plataformas UNIX.

[4] Una *mailing list* o listas de distribución son foros de discusión privados a través de correo electrónico. Las listas de correo están formadas por direcciones e-mail de los usuarios que la componen. Cuando uno de los participantes envía un mensaje a la lista, ésta reenvía una copia del mismo al resto de usuarios de la lista (inscritos en ella).

6.2.2.2 Nxlog

El sistema Nxlog en la versión que hemos estado utilizando durante el trabajo (la versión Nxlog Community Edition, la cual es Open Source), igual que ocurre en SEC, es un sistema que está orientado tanto a las pequeñas como medianas empresas. Nxlog, además de esta versión, tiene una versión comercial la cual está orientada a empresas con necesidades de alta carga.

Nxlog es una herramienta que al principio parece que puede resultar un poco tediosa de usar. Pero una vez familiarizado con los módulos, su uso es muy parecido al de SEC ya que el modulo de correlación está basado en él.

6.2.2.2.1 Ventajas

Algunas ventajas son:

- No solo realiza correlación, sino que también ayuda a la conversión en otros formatos, para poderlo llevar a otras plataformas si fuera necesario. Lo que conlleva a poder leer casi todo tipo de formatos de log (Syslog, JSON…).
- Tiene también un soporte como el de Nxlog, pero no es tan eficiente. Puede recibir la respuesta a los 10 días después de haberla formulado.
- Tiene funcionalidades mejoradas con respecto a SEC. Tal y como ya hemos comentado en otros apartados, Nxlog está escrito en C y según el creador da un rendimiento mayor. Lo cual ya hemos comprobado que es cierto en los resultados del experimento.
- Este software funciona en el 90% de plataformas, hasta en sistemas Android.

6.2.2.2.2 Inconvenientes

Algunos de los inconvenientes que hemos observado son:

- Aunque, en el apartado anterior hemos comentado que ofrece servicio en casi todas las plataformas, tiene limitación en la instalación. Ya que, para el sistema operativo Ubuntu, solo proporciona los paquetes para los dispositivos con un procesador AMD.
- Aprendizaje de uso de cada uno de los módulos para poner en marcha el sistema.

6.2.2.3 Prelude

Prelude no es solamente un correlador, está formado por diferentes softwares de ahí a que no sea únicamente un sistema correlador si no que es considerado como un sistema SIEM. Ya que ofrece, entre otras cosas, un *dashboard* [5], en este caso es Prewikka.

Igual que en el caso de Nxlog, Prelude posee una versión comercial la cual nos ofrece mucha más características, la herramienta que nosotros estamos usando es la versión Open Source, destinada a la educación o test de funcionamiento. Lo que conlleva a sus desarrolladores a limitar este producto.

6.2.2.3.1 Ventajas

Algunas ventajas son:

- Como ya hemos comentado, la posesión de un *dashboard.*
- Uso de un motor de correlación mucho más avanzado, que en el caso de SEC y Nxlog, basado en reglas de Python.
- Compatibilidad con una gran cantidad de sensores como Snort, Suricata, etc…
- La posibilidad de hacer compatible otros sensores, que aún no lo son, gracias a unas librerías que se proporcionan.

[5] Un *dashboard* es una herramienta que puede tomar los datos de eventos y convertirlo en gráficos informativos para ayudar a ver patrones, o la identificación de actividad que no está formando un patrón estándar.

6.2.2.3.2 Inconvenientes

Algunos de los inconvenientes que hemos observado son:

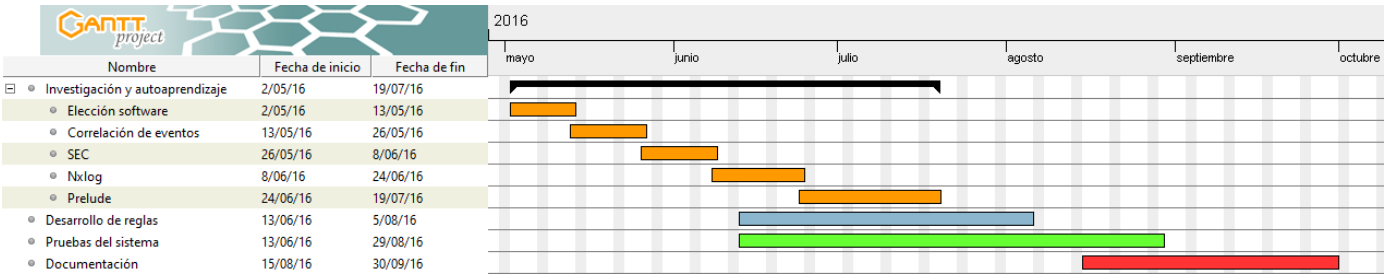
- Solo ofrece correlación en tiempo real.
- No se ofrece ningún manual de uso, como en SEC y Nxlog. Únicamente, se tiene la poca información que ofrece la web de Prelude o por autoaprendizaje.
- La instalación es bastante compleja si se quiere hacer uso del sistema en un entorno personal.
- Una de las opciones que proporciona cualquier correlador para el aviso al supervisor de la red, es el aviso por email. En este caso no se proporciona y lo cual es un gran fallo. Ya que, si se considera un fallo de nivel alto, lo primero que se hace es notificar al gestor para que esté informado.

7 PLANIFICACIÓN

Do all you can to make your dreams come true.

- Joel Osteen-

En este capítulo mostramos mediante la figura 6-1, un diagrama de Gantt en el que se describe, de forma general, la planificación seguida a lo largo de este trabajo. Como se puede observar, en el diagrama se muestran las tareas más generales que se han llevado a cabo clasificadas en varias secciones: investigación, desarrollo, pruebas y documentación.



Nombre	Fecha de inicio	Fecha de fin
Investigación y autoaprendizaje	2/05/16	19/07/16
Elección software	2/05/16	13/05/16
Correlación de eventos	13/05/16	26/05/16
SEC	26/05/16	8/06/16
Nxlog	8/06/16	24/06/16
Prelude	24/06/16	19/07/16
Desarrollo de reglas	13/06/16	5/08/16
Pruebas del sistema	13/06/16	29/08/16
Documentación	15/08/16	30/09/16

Figura 7-1. Diagrama de Gantt del trabajo

8 CONCLUSIONES

The most important property of a program is whether it accomplishes the intention of its user.

- C.A.R. Hoare -

En está memoria se ha intentado reflejar todo el trabajo que se ha llevado a cabo durante varios meses de aprendizaje, investigación y trabajo. Durante todo este periodo he podido aprender como funcionan y se utilizan los sistemas de correlación. Una tecnología que está siendo actualmente ampliamente utilizada en la gestión de redes.

Una vez terminado el trabajo, se puede decir que se ha conseguido realizar una comparación y poner en marcha los sistemas de correlación en el sistema operativo Ubuntu, además de configurar sus archivos de reglas para que correlen una serie de eventos.

Después de todo el trabajo realizado tengo la sensación de haber cumplido todos los objetivos que inicialmente se habían propuesto, incluso han ido aumentándose mis expectativas en la medida que iba aprendiendo elementos que desconocía, como el lenguaje Python o el uso de expresiones regulares.

Todo ello me produce una gran satisfacción de haber realizado un buen trabajo.

8.1 Líneas de continuación

De cara a un desarrollo futuro, una mejora en los sistemas de correlación que hemos visto sería la integración de una plataforma con interfaz gráfica para la gestión de las reglas. Lo cual abriría más ampliamente su campo de ventas o uso, ya sean sistemas comerciales o no. Ya que, podría ser utilizado en entornos personales.

Otra posibilidad sería la de desarrollar estos softwares para sistemas como Windows o MacOS ya que siempre son los grandes olvidados en estos tipos de servicios. Y junto a la anterior mejora podrían evolucionar bastante.

TRABAJOS CITADOS

[1] A. Rodríguez, *Sistema de Gestión Syslog en Cloud.,* 2016.

[2] A. Müller, *Event Correlation Engine,* 2009.

[3] A. Martinez, «Platzi,» 2015. [En línea]. Available: https://platzi.com/blog/expresiones-regulares-python/.

[4] R. Vaarandi, «SEC manual page,» 2016. [En línea]. Available: https://simple-evcorr.github.io/man.html.

[5] B. Botyanszki, *NXLOG Community Edition Reference Manual.*

GLOSARIO

IDMEF: Intrusion Detection Message Exchange Format 21

IP: Internet protocol 13

LML: Log Monitoring Lackey 21

MARS: Monitoring, Analysis, and Response System 32

SEC: Simple Event Correlator 2

SIEM: Security Information and Event Management 17

TCP: Transmission Control Protocol 17

UDP: User Datagram Protocol 17

ANEXO A: INSTALACIÓN SEC

En este anexo, mostraremos paso por paso el proceso para instalar el sistema correlador SEC.

1. Entramos en el terminal en modo superusuario con la siguiente orden:

```
su -
```

2. Nos descargamos el paquete desde la web de SEC:

```
wget http://sourceforge.net/projects/simple-
evcorr/files/sec/2.7.10/sec-2.7.10.tar.gz
```

3. Descomprimimos el paquete:

```
tar -zxf sec-2.7.10.tar.gz
```

El paquete contiene entre otros elementos, el script sec.pl que es el motor de correlación y el manual.

4. Copiamos, el script a su correcta localización:

```
cd sec-2.7.10

cp sec /usr/local/sbin
```

5. Ponemos en funcionamiento el sistema:

```
/usr/local/sbin/sec -conf=/home/antonio/sec-2.7.8/config/L01.conf
-input=/home/antono/TFG/entrada -log=/home/antono/TFG/salida
```

En el cual le indicamos:

- El archivo de configuración, donde se encuentran las reglas.
- El archivo de donde vamos a leer los logs. Si queremos que la entrada sea el propio terminal lo indicamos con un guion: -input= -.
- Y el archivo donde queremos que nos muestre la salida del programa.

ANEXO B: INSTALACIÓN NXLOG

En este anexo, mostraremos paso por paso el proceso para instalar el sistema correlador Nxlog.

1. Entramos en modo superusuario con la siguiente orden:

```
su -
```

2. Nos descargamos el paquete desde la web de Nxlog:

```
wget https://nxlog.co/system/files/products/files/1/nxlog-ce-
2.8.1248.tar.gz
```

3. Descomprimimos el paquete:

```
tar -zxf nxlog-ce-2.8.1248.tar.gz
```

4. Instalamos una serie de librerías y paquetes necesarios para su instalación:

```
apt-get install libpcre3-dev libapr1-dev libssl-dev libexpat-dev
make
```

5. Procedemos a su instalación:

```
./configure

make

make install
```

6. Realizamos las siguientes acciones antes de poner en funcionamiento el sistema:

```
mkdir -p /usr/local/var/run/nxlog/

mkdir /var/log/nxlog/

mkdir -p /usr/local/var/spool/nxlog/

mkdir /usr/local/etc/nxlog

useradd nxlog

cp /root/nxlog-ce-2.8.1248/packaging/debian/nxlog.init
/etc/init.d/nxlog

sed -i 's/\/usr\/bin\/nxlog/\/usr\/local\/bin\/nxlog/g'
/etc/init.d/nxlog
```

Una vez realizada estas acciones, hay que introducir el fichero de configuración en */etc/nxlog/nxlog.conf*

Las opciones que nos das Nxlog para iniciarlo son:

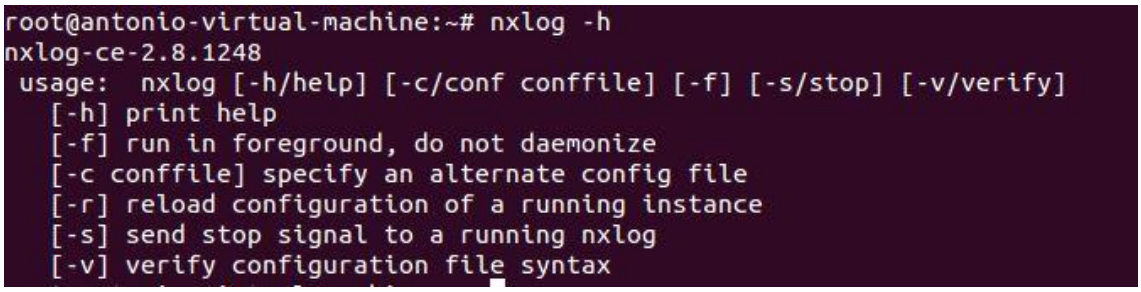


Figura B-1. Opciones Nxlog

7. Ponemos en funcionamiento el sistema:

```
nxlog
```

8. Para que entren en funcionamiento nuevas reglas después de haberlo iniciado, realizamos la siguiente acción:

```
nxlog -r
```

El cual manda una señal de SIGHUP[6] al proceso en ejecución.

[6] La señal HUP hace que el proceso padre (cierre los hijos y) relea sus ficheros configuración (y logs), sin terminarlo.

ANEXO C: INSTALACIÓN PRELUDE

En este anexo, mostraremos paso por paso el proceso para instalar el sistema correlador Prelude. Para que sea más fácil de entender nos apoyaremos en múltiples capturas del proceso.

C.1 Instalación del manager

1. Entramos en el terminal en modo superusuario con la siguiente orden:

```
su -
```

2. Actualizamos nuestro sistema de gestión de paquetes, para obtener la última versión de nuestro software, con la siguiente orden:

```
apt-get update
```

3. Instalamos el manager de Prelude:

```
apt-get -y install prelude-manager
```

4. Nos preguntará si queremos configurar una base de datos con la ayuda de dbconfig-common. En el caso de que tengamos ya una preinstalada pulsamos que no y configuramos. En nuestro pulsamos que sí, porque no tenemos ninguna:

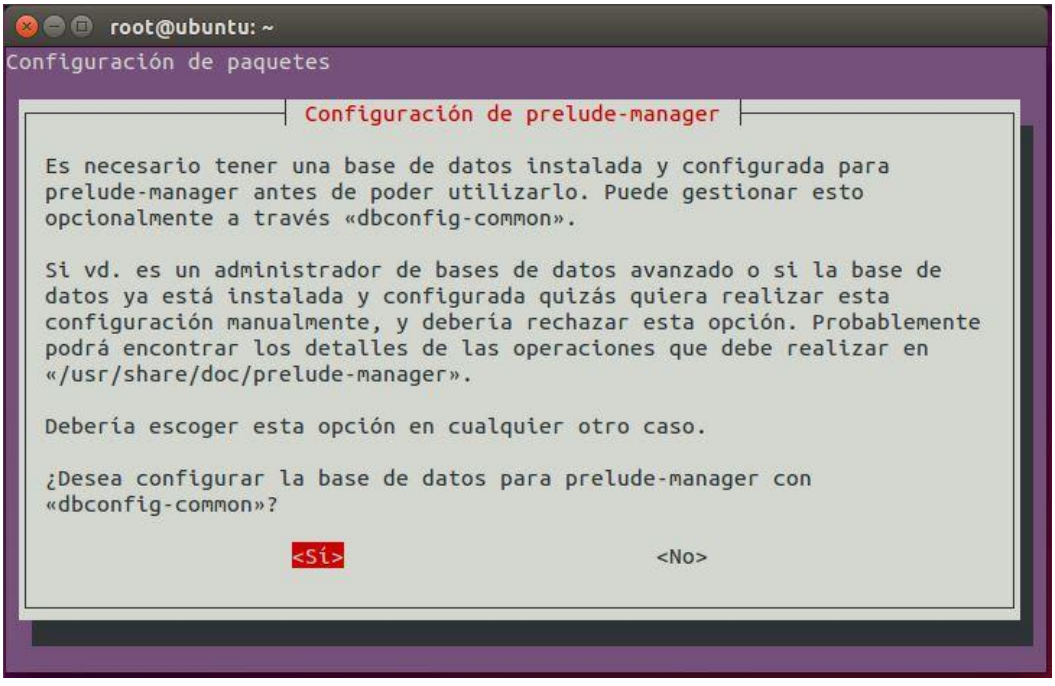


Figura C-1. Petición configuración base de datos

5. Nos preguntara que base de datos quiere que instalemos, elegimos MySQL, aunque podría ser también la otra opción:

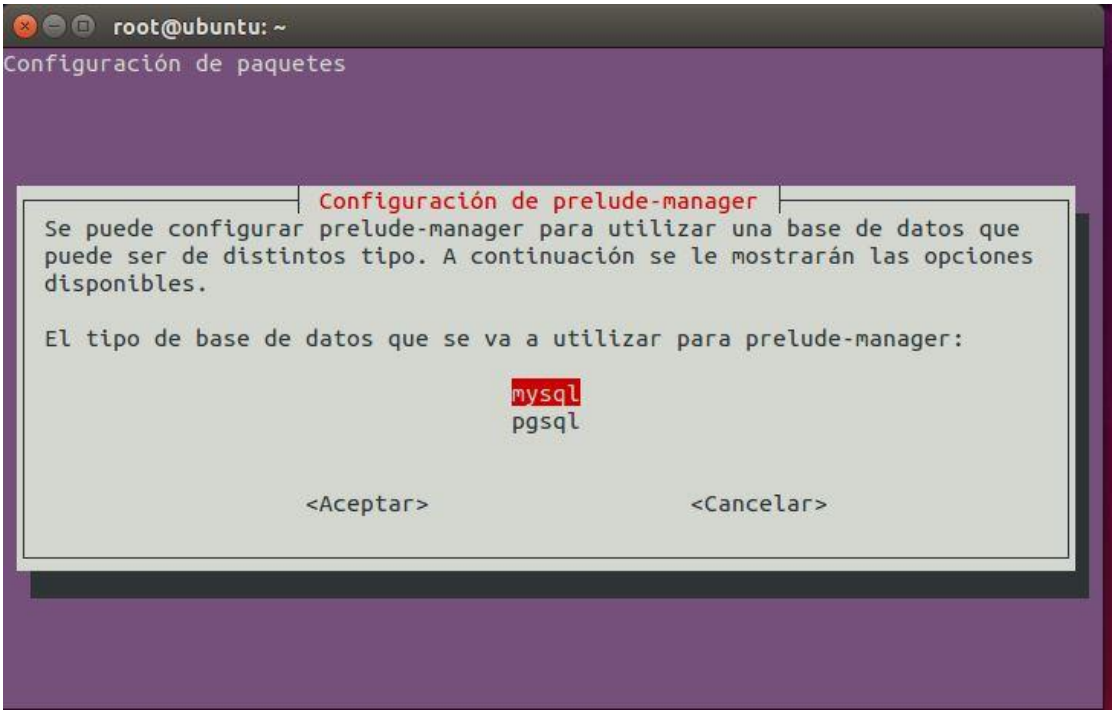


Figura C-2. Elección de base de datos

6. Nos pedirá una contraseña para gestionar la base de datos, en nuestro caso hemos usado root:

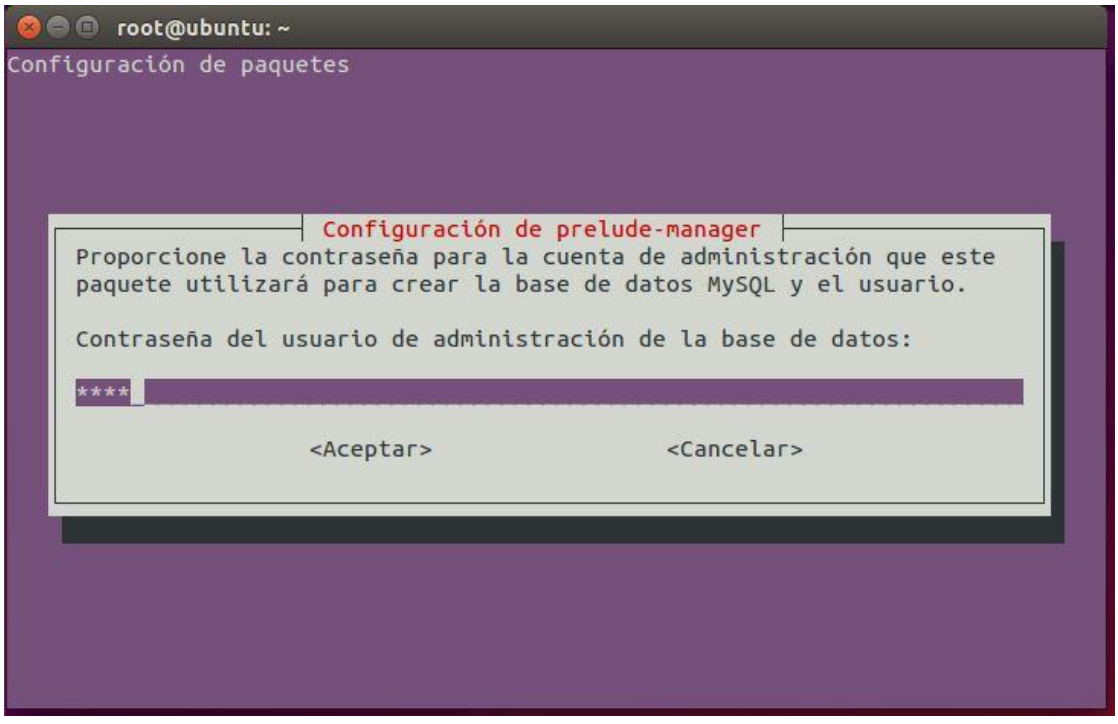


Figura C-3. Petición de contraseña para el root

7. Nos pedirá a continuación la contraseña para la base de datos del manager, hemos usado también root:

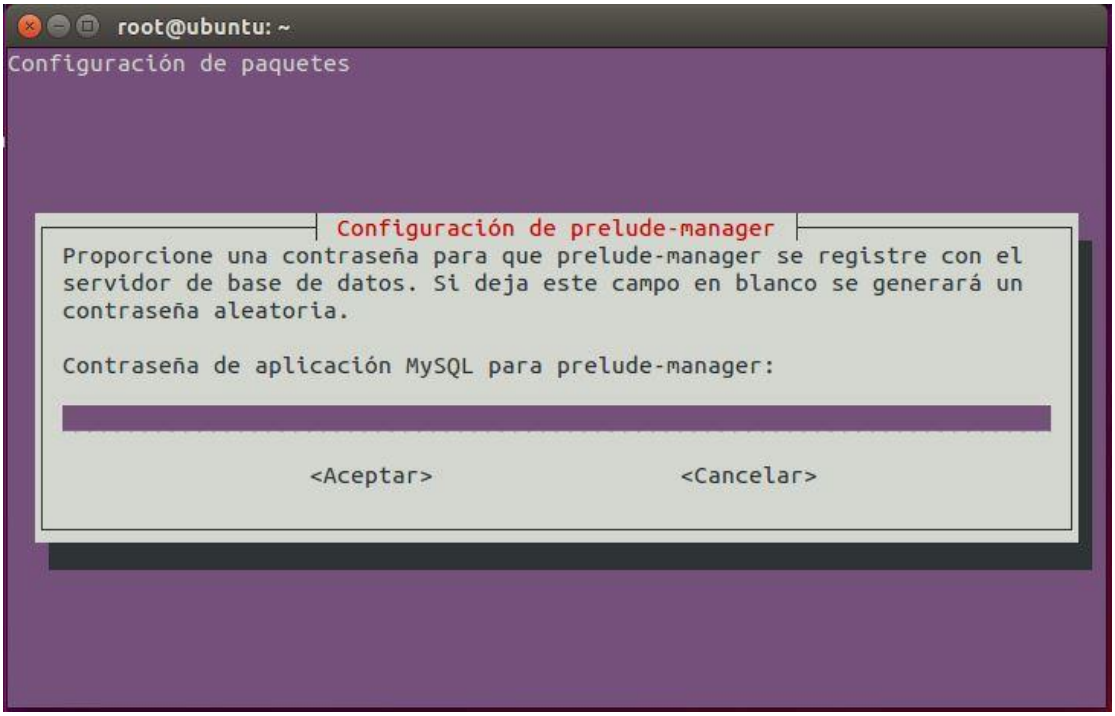


Figura C-4. Petición de contraseña para la base de datos del manager

8. Repetimos la clave para confirmarla:

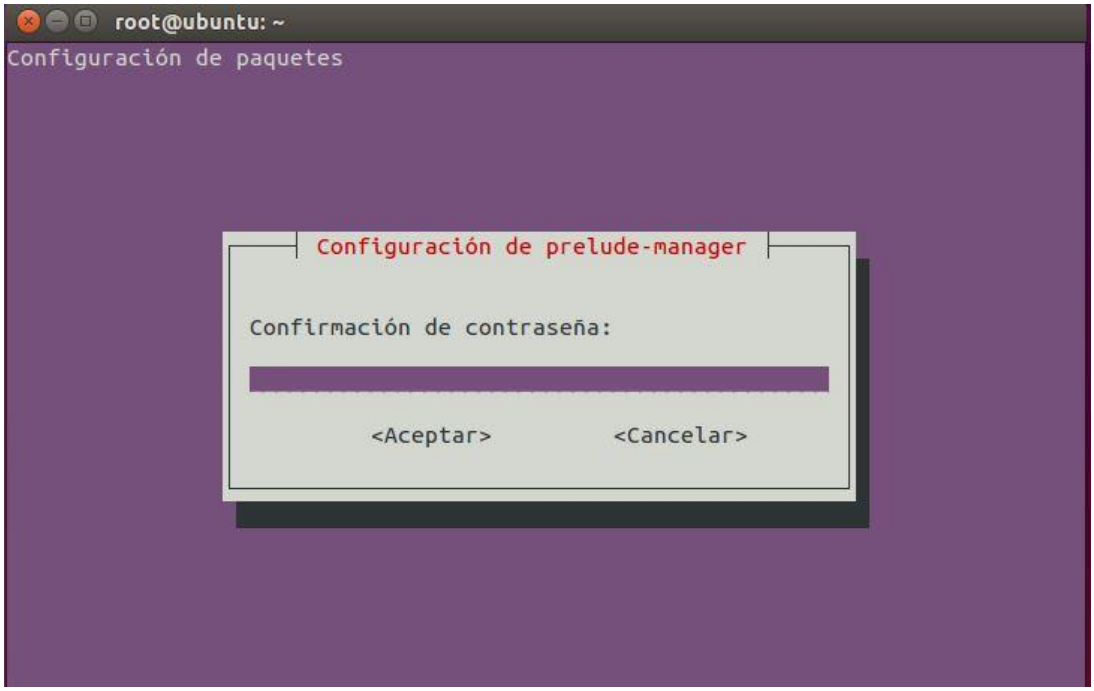


Figura C-5. Confirmación de contraseña para la base de datos del manager

9. Una vez haya terminado la instalación hay que modificar el archivo */etc/default/prelude-manager* y establecer el valor de RUN a yes.

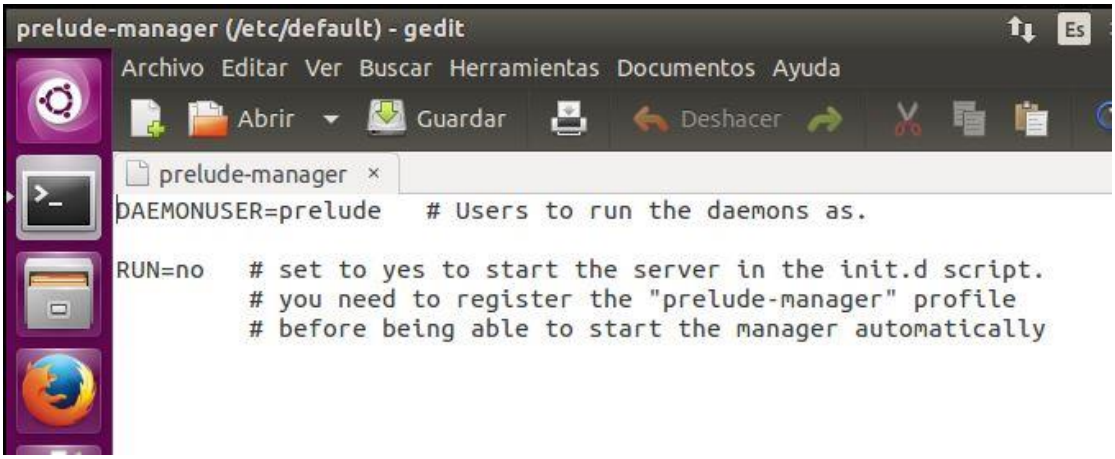


Figura C-6. Archivo */etc/default/prelude-manager*

10. Ejecutamos Prelude-manager:

```
/usr/sbin/prelude-manager -d -P /var/run/prelude-manager.pid
```

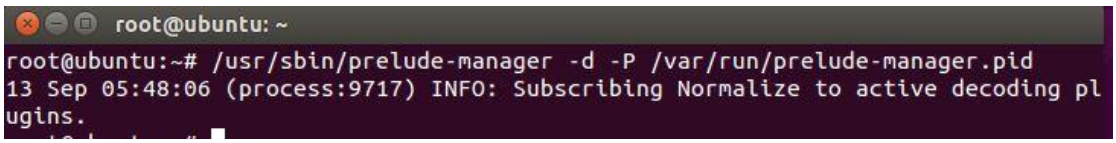


Figura C-7. Ejecución de Prelude-manager

C.2 Instalación del correlador

1. Instalamos Prelude Correlator:

```
apt-get install prelude-correlator
```

2. Registramos Prelude-Correlator para crear el perfil en Prelude-Manager. En una terminal (A) hacemos lo siguiente:

```
prelude-admin registration-server prelude-manager
```

Nos proporcionará la contraseña que en nuestro caso es: *1fliefg8* la cual debemos introducir en el registro del correlador.

Figura C-8. Registro en Prelude-manager

3. Para registrar el correlador realizamos lo siguiente en un terminal (B):

```
prelude-admin register prelude-correlator "idmef:rw" 127.0.0.1 -
uid 0 -gid 0
```

4. Nos aparecerá dos campos donde debemos introducir dicha clave:

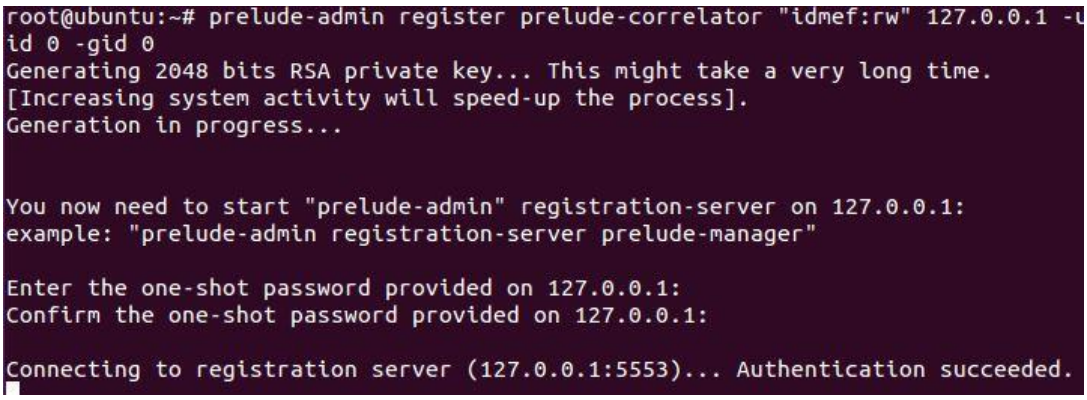


Figura C-9. Registro del Prelude Correlator

5. En el terminal (A) nos aparecerá que confirmemos:

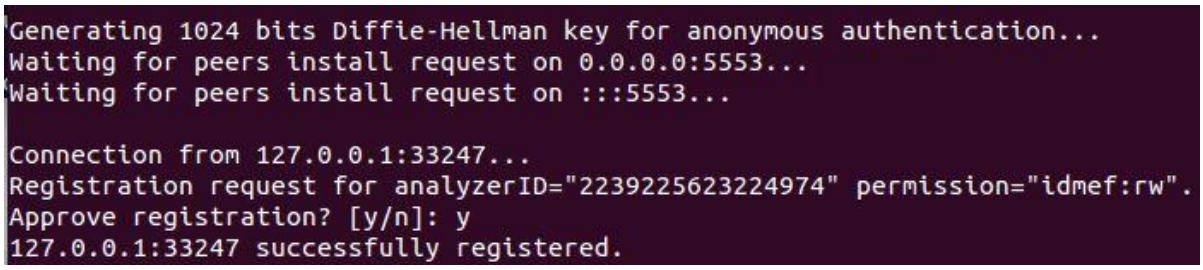


Figura C-10. Registro correcto en el manager

6. Ejecutamos el correlador:

```
prelude-correlator
```

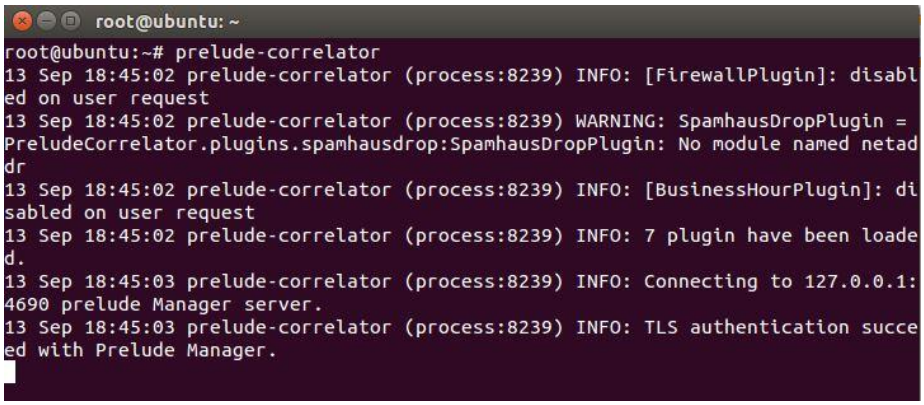


Figura C-11. Arranque del correlador

7. Comprobamos que se ha registrado correctamente introducimos:

```
prelude-admin list -l
```

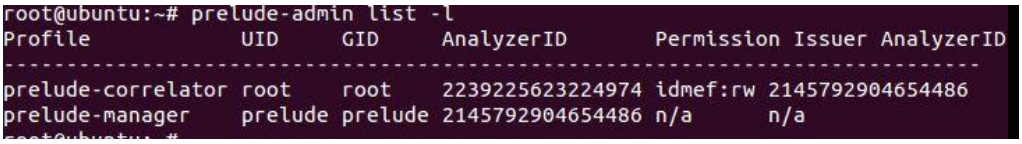


Figura C-12. Comprobación de registro

Si queremos sacar la información que correla, ejecutamos el correlador de la siguiente manera:

```
prelude-correlator --print-output=ARCHIVO
```

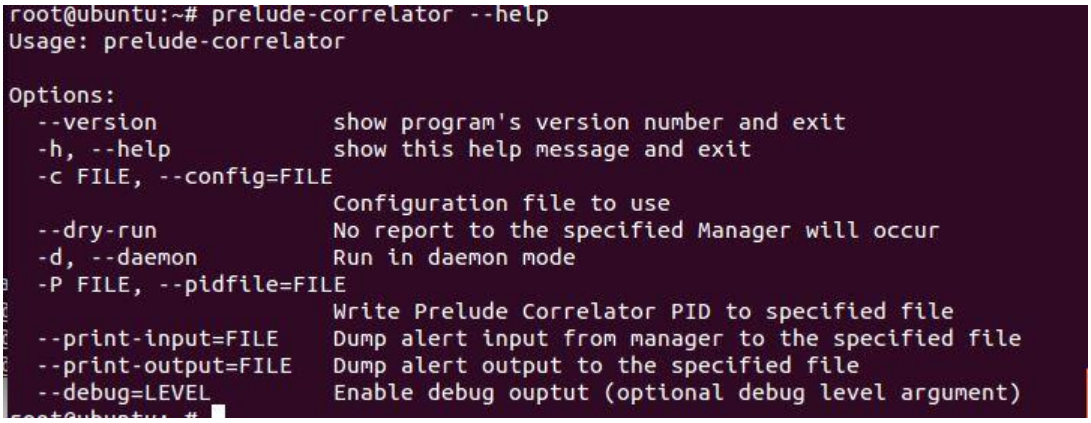


Figura C-13. Opciones ejecución Prelude Correlator

El archivo de configuración del correlador lo podemos encontrar en */etc/prelude-correlator/prelude-correlator.conf* en el cual podemos activar y desactivar las reglas o plugins que se encuentran en */usr/share/pyshared/PreludeCorrelator/plugins*

De esta manera ya tenemos en funcionamiento nuestro correlador. Pero nos hace falta ahora un sensor para detectar los eventos.

C.3 Instalación de Prewikka

Primeramente, procedemos a la instalación de Prewikka para tener una mejor visualización de lo que ocurre.

1. Instalamos ahora el paquete de Prewikka. Se nos pedirá que lo registremos en la base de datos, igual que Prelude Correlator. Introduciremos las respectivas contraseñas, en nuestro caso root.

```
apt-get install prewikka
```

2. Ejecutamos en segundo plano Prewikka:

```
prewikka-httpd &
```

3. Abrimos un navegador e introducimos:

```
localhost:8000
```

Y visualizamos esto:

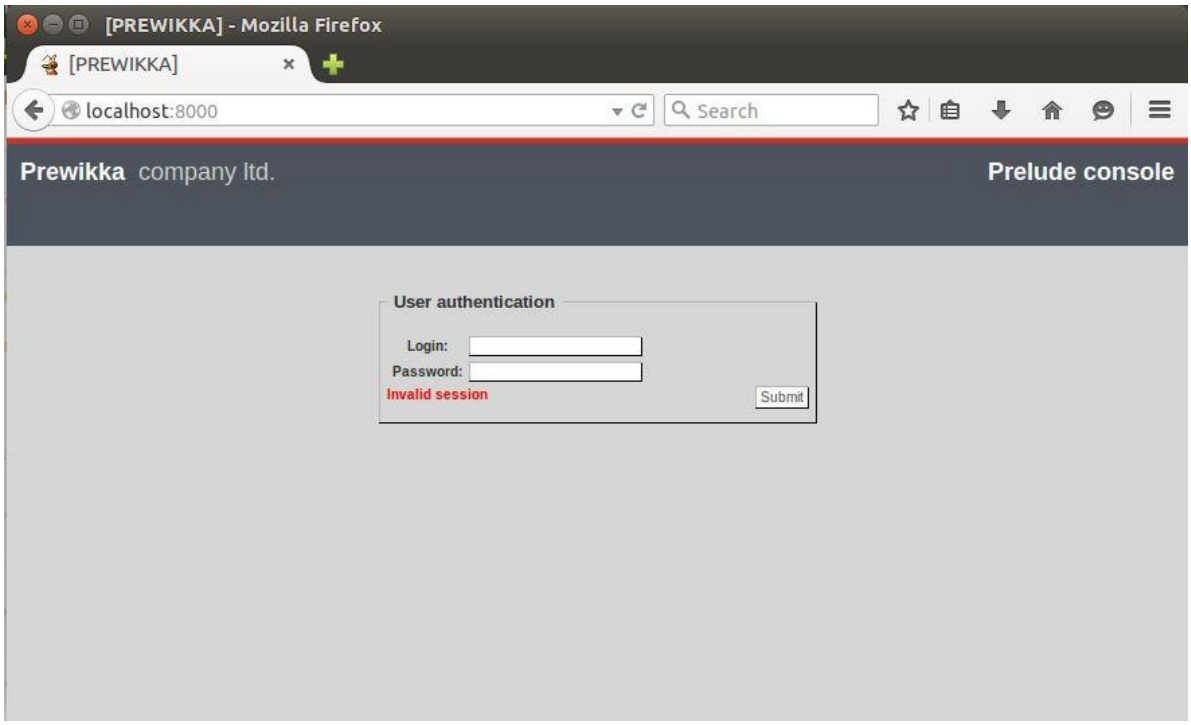


Figura C-14. Página de inicio de sesión Prewikka

En este lugar debemos introducir como usuario y contraseña: admin/admin.

Vemos ahora los sensores que tenemos activos:

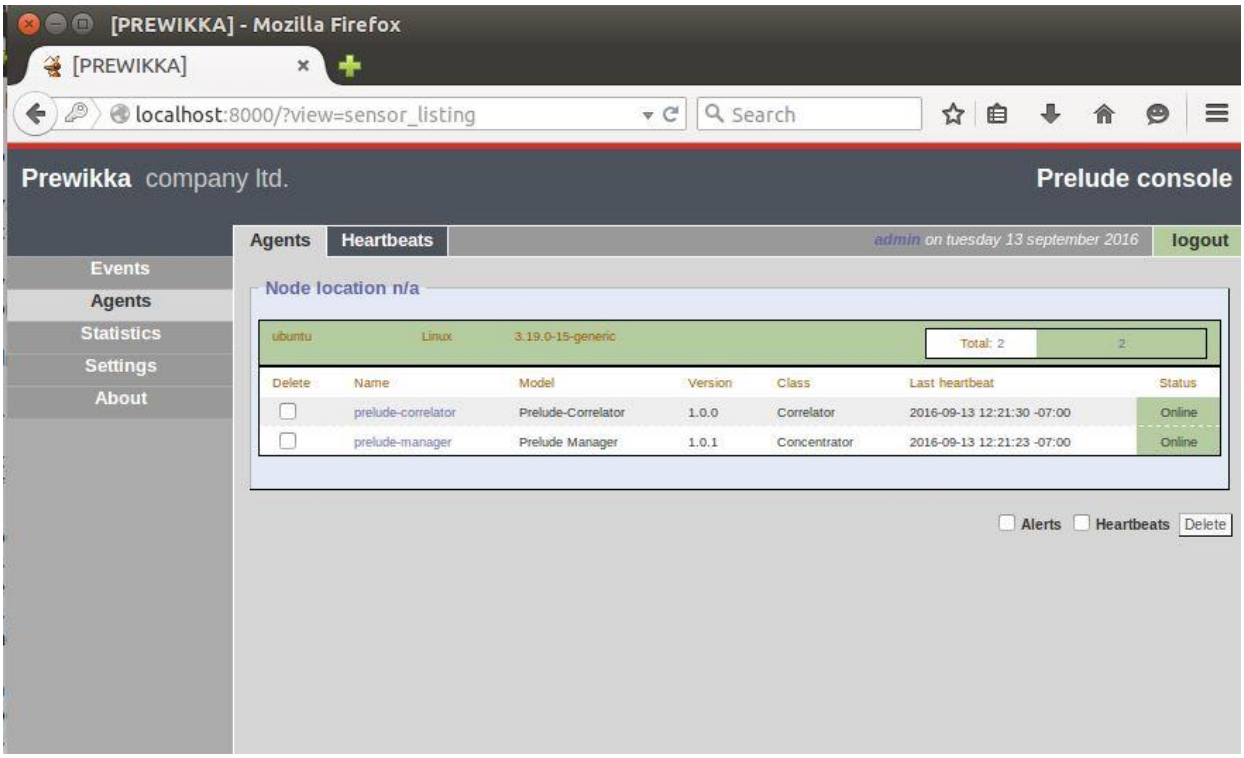


Figura C-15. Sensores en Prewikka

C.4 Instalación de Snort

Ahora procedemos a la instalación del sensor Snort.

1. Instalamos una serie de requisitos para Snort:

```
  apt-get install libnet1 libnet1-dev autoconf libtool gcc-4.4 g++
automake gcc make flex bison libpcre3 libpcre3-dev
```

También necesitaremos Libpcap y Libdnet. Para ello procederemos con las siguientes acciones.

2. Instalación de Libpcap:

```
  wget http://www.tcpdump.org/release/libpcap-1.1.1.tar.gz

  tar -zxf libpcap-1.1.1.tar.gz

  cd libpcap-1.1.1

  ./configure -prefix=/usr -enable-shared

  make

  make install
```

3. Instalación de Libdnet:

```
wget ftp://ftp.netbsd.org/pub/pkgsrc/distfiles/libdnet-1.12.tgz

tar -zxf libdnet-1.12.tgz

cd libdnet-1.12

./configure -prefix=/usr -enable-shared

make

make install
```

4. Instalación del daq:

```
wget https://www.snort.org/downloads/snort/daq-2.0.6.tar.gz

tar zxvf daq-2.0.6.tar.gz

cd daq-2.0.6

./configure

make

make install

ldconfig
```

5. Instalamos las librerías necesarias para el registro de sensores en Prelude:

```
apt-get install libgnutls-dev python lua50 libprelude-dev
libpreludedb-dev
```

6. Instalamos Snort, no instalamos la última versión (2.9.8.3) porque aún no tiene soporte para Prelude:

```
wget
http://downloads.sourceforge.net/project/snort/OLD%20STUFF%20THAT%2
0YOU%20SHOULDNT%20USE/snort-2.9.0.4.tar.gz

tar -zxf snort-2.9.0.4.tar.gz

cd snort-2.9.0.4

./configure -enable-prelude -enable-dynamicplugin -enable-zlib -
enable-reload -enable-ipv6

make

make install
```

7. Comprobamos que funciona correctamente:

```
snort -V
```

8. Creamos las carpetas necesarias para alojar las reglas y archivos de configuración:

```
mkdir /etc/snort

mkdir /etc/snort/rules

mkdir /var/log/snort

mkdir /etc/snort/proproc_rules

mkdir /etc/snort/so_rules

mkdir /etc/snort/preproc_rules
```

9. Descargamos las reglas:

```
wget
http://downloads.sourceforge.net/project/douglashx/snort/snortrules
-snapshot-2904.tar.gz

tar zxvf snortrules-snapshot-2904.tar.gz
```

10. Pasamos la información descargada a cada una de las carpetas correspondientes creadas en el punto 8 y modificamos */etc/snort/snort.conf* según nuestras necesidades, como activar preprocesadores como sfportscan, activar preproc_rules, etc…
11. En */etc/snort/snort.conf*, modificamos la línea "*# output alert_prelude"* por "*output alert_prelude : profile=snort-1"*. En profile se puede poner el nombre que se quiera, en este caso le hemos añadido un 1 porque podemos tener más sensores Snort distribuidos en otros sistemas.
12. Registramos Snort en el sistema, igual que antes el correlador:

```
prelude-admin registration-server prelude-manager

prelude-admin register "snort-1" "idmef:w" 127.0.0.1 –uid 0 –gid
0
```

13. Iniciamos Snort:

```
snort -ieth0 -dev -c /etc/snort/snort.conf
```

Vemos su correcta inicialización:

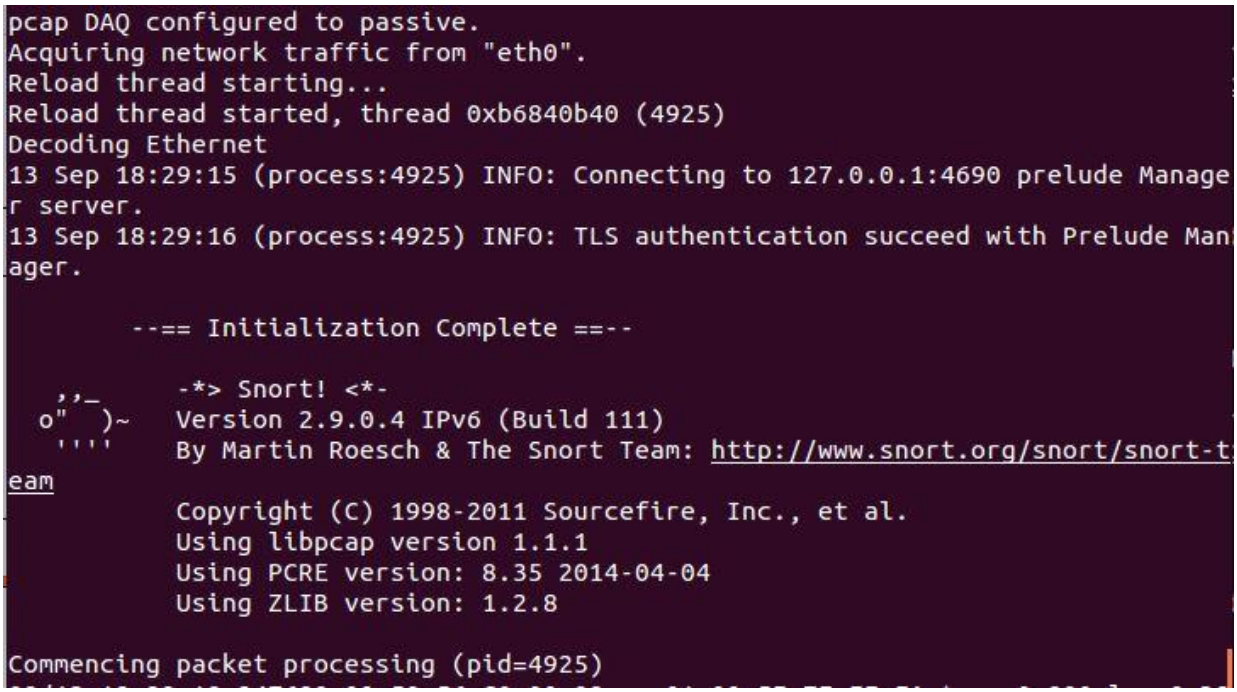


Figura C-16. Inicialización Snort

14. Vemos en Prewikka si está activo el sensor:

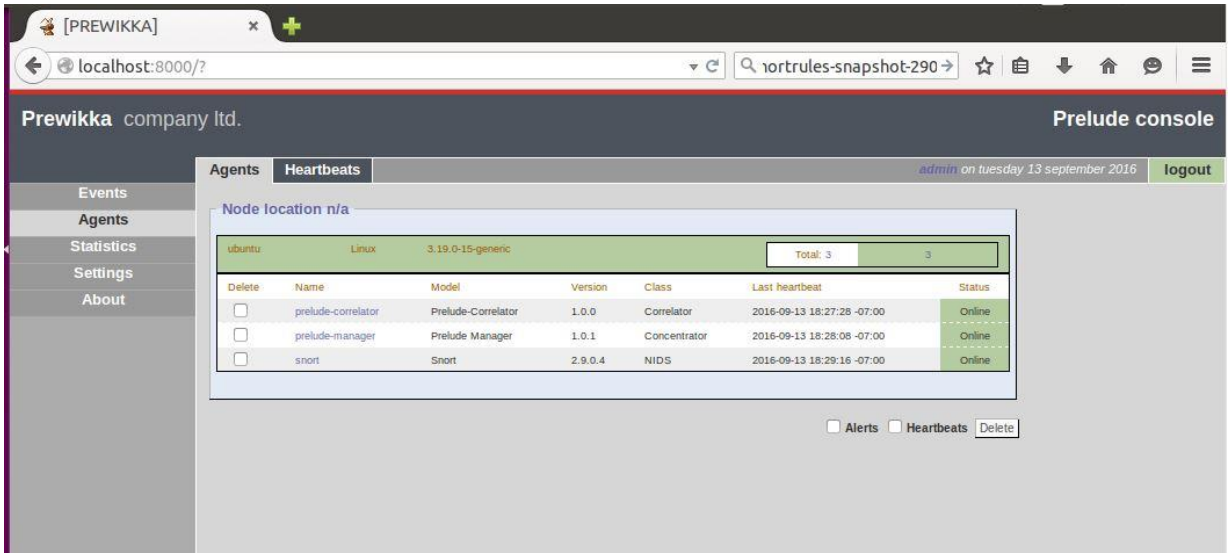


Figura C-17. Sensores activos después de instalar Snort

C.5 Comprobación funcionamiento

1. Realizamos una última comprobación con el envío de 9 pings:

```
root@ubuntu:~# ping google.es
PING google.es (216.58.211.227) 56(84) bytes of data.
64 bytes from mad01s24-in-f227.1e100.net (216.58.211.227): icmp_seq=1 ttl=128 time=44.4
 ms
64 bytes from mad01s24-in-f227.1e100.net (216.58.211.227): icmp_seq=2 ttl=128 time=115
ms
64 bytes from mad01s24-in-f227.1e100.net (216.58.211.227): icmp_seq=3 ttl=128 time=58.8
 ms
64 bytes from mad01s24-in-f227.1e100.net (216.58.211.227): icmp_seq=4 ttl=128 time=123
ms
64 bytes from mad01s24-in-f227.1e100.net (216.58.211.227): icmp_seq=5 ttl=128 time=74.6
 ms
64 bytes from mad01s24-in-f227.1e100.net (216.58.211.227): icmp_seq=6 ttl=128 time=129
ms
64 bytes from mad01s24-in-f227.1e100.net (216.58.211.227): icmp_seq=7 ttl=128 time=142
ms
64 bytes from mad01s24-in-f227.1e100.net (216.58.211.227): icmp_seq=8 ttl=128 time=220
ms
64 bytes from mad01s24-in-f227.1e100.net (216.58.211.227): icmp_seq=9 ttl=128 time=364
ms
^C
--- google.es ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 8282ms
rtt min/avg/max/mdev = 44.433/141.518/364.789/93.123 ms
```

Figura C-18. Envío de 9 pings

2. Vemos si ha sido capturado en el manager y mostrado en Prewikka:

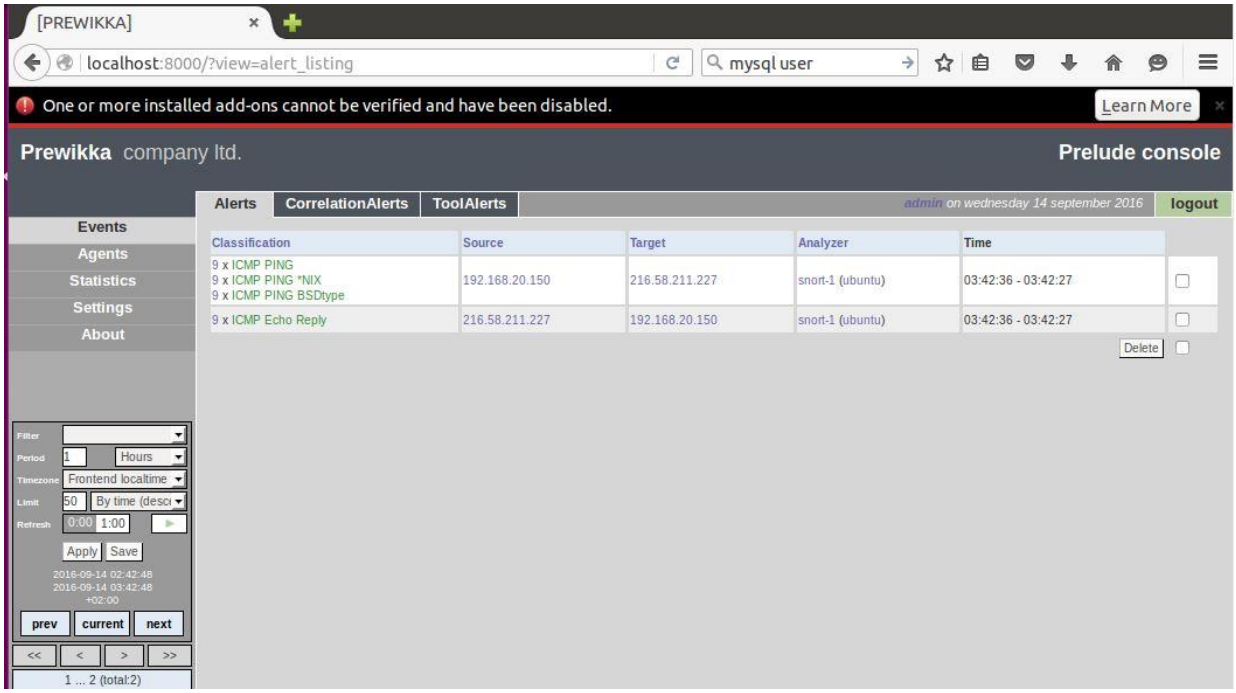


Figura C-19. Visualización de la detección del ping en Prewikka