DETECCIÓN DE EXTRACCIÓN NO AUTORIZADA DE INFORMACIÓN CON SNORT

Proyecto postgrado seguridad en sistemas y redes

Memoria
6 de junio de 2016

Autor: Andrés Adrover Llinás
Tutora: Cristina Pérez Solà

Agradecimientos

Este trabajo está dedicado a todas aquellas horas de sueño que robamos para dedicarlas a aprender; a estrellarnos, una y otra vez, contra una configuración que se resiste; a rastrear entre la documentación en busca del porqué de un fallo; o simplemente a darle vueltas a la cabeza para ver cómo plasmar por escrito lo aprendido.

Agradecer la paciencia de mi familia durante este año en el que mi presencia en la vida familiar se ha reducido de forma importante al pasarme los fines de semana enganchado al ordenador entre prácticas y estudio.

No puedo dejar de agradecer los consejos y guía de mi tutora: Cristina Pérez Solà.

Licencia

Resumen

El objetivo de este trabajo es la presentación de una prueba de concepto que demuestre las posibilidades, así como los requisitos y configuraciones necesarias, del software Snort para la detección de una posible extracción no autorizada de información empresarial.

La necesidad de este proyecto surge por una mala experiencia previa de la empresa solicitante de la consultoría que implicó la extracción no autorizada de información confidencial de la empresa por parte de unos de los anteriores técnicos de sistemas.

En este documento revisaremos los requisitos previos y las premisas que debemos tener en cuenta para obtener una prueba de concepto que muestre las posibilidades reales de obtener un producto que cubra las necesidades expresadas por el cliente.

Una vez definidas y explicadas las premisas mostraremos las principales características de la solución técnica propuesta, tanto en lo que a las firmas de detección se refiere como a diferentes configuraciones que nos permitirán obtener un producto final acorde con las necesidades del cliente.

La solución propuesta orbita sobre la definición como conductas sospechosas de una posible extracción no autorizada de información al hecho de que aparezcan diferentes conexiones fuera del horario laboral. Se ha definido esta diferenciación horaria ya que es técnicamente imposible diferenciar los accesos legítimos y necesarios de aquellos que tienen fines no autorizados. Esta diferenciación horaria implica un trabajo de configuración específico para que el sistema active o desactive las firmas de detección de conexiones posiblemente malintencionadas según los requerimientos de la empresa.

Contenido

Índice de tablas

Índice de ilustraciones

1. Introducción

Este proyecto surge como respuesta a una necesidad específica de una empresa no tecnológica de un tamaño medio: la detección de extracción no autorizada de información.

Tras un despido no amigable de uno de los técnicos informáticos se detectó la extracción no autorizada de información empresarial por lo que se solicita una prueba de concepto de un sistema que permita la detección de una posible extracción de información de información.

Para evitar la repetición de este suceso la empresa solicitó la viabilidad tecnológica de un sistema que detectase este comportamiento a su departamento técnico, pero ante la falta de conocimientos y del tiempo necesario para realizar una investigación propia se solicita la asistencia técnica externa para realizar este proyecto.

Al tratarse de una empresa no tecnológica y que tan sólo cuenta con dos técnicos informáticos para realizar todas las tareas de instalación, configuración y mantenimiento de la infraestructura tecnológica de la empresa se solicita que la gestión y revisión de los posibles eventos y alarmas no sea costosa en tiempo.

Durante los contactos iniciales acordamos realizar únicamente una prueba de concepto para poder demostrar la viabilidad técnica y las capacidades reales de detección de estos comportamientos antes de pensar en una implementación real en la infraestructura tecnológica de la empresa. Tras esta prueba de concepto tanto los técnicos propios como la dirección de la empresa evaluarán la idoneidad del sistema para una instalación en real.

1.1. Definición del problema

Tal como hemos mencionado en la introducción el problema que busca resolver este proyecto es la detección, y consiguiente generación de eventos y alarmas, de comportamientos anómalos que puedan sugerir la extracción no autorizada de información privada de la empresa.

Así nuestra propuesta debe ser un sistema autónomo, que implique poca carga de trabajo para los técnicos de la empresa, que permita la detección de comportamientos anómalos que induzcan a pensar en una posible extracción no autorizada de información desde los diferentes entornos informáticos con los que cuenta la empresa.

1.2. Objetivos

Los objetivos que debe cubrir el proyecto se basan en las necesidades específicas de esta empresa: la detección de una posible extracción no autorizada de información privada de la empresa. No se trata de una instalación genérica, o teórica, de un sistema IDS/IPS sino de una configuración específica para cumplir la necesidad concreta de esta empresa.

Ya que este proyecto se basa en la presentación de una prueba de concepto que permita demostrar las capacidades del sistema para cubrir las necesidades específicas del cliente nos basaremos en ellas para realizar una configuración del sistema Snort que permita dar la respuesta deseada.

Así para cubrir los requisitos demandados preparemos un sistema que cubra los tres objetivos principales del cliente:

- Detección de comunicaciones anómalas que puedan implicar una extracción de información no autorizada.

Se buscarán localizar comportamientos anómalos en las comunicaciones de los servidores corporativos. Por regla general los servidores reciben conexiones desde los clientes, pero no suelen ser los iniciadores de conexiones al exterior por lo que la detección de una conexión hacia el exterior cuyo origen sean dichos servidores puede ser un indicio importante de una extracción de información. Seleccionaremos o crearemos las firmas de detección oportunas para estos casos.

- Registro de eventos y alarmas
  Debido a la experiencia previa se solicita que los eventos y alarmas que puedan generarse queden debidamente registrados para su posterior análisis, aunque el usuario malicioso fuera uno de los técnicos informáticos. Para solventar esta posibilidad deberemos configurar el sistema Snort para que guarde una copia de los eventos en un servidor externo.
- Minimización de tiempos de gestión
  El departamento técnico, cuya prioridad absoluta es la resolución de incidencias, tienen una alta carga de trabajo por lo que la propuesta deberá permitir que la adición de esta nueva tarea no suponga un consumo excesivo de tiempo. Así presentaremos un sistema de gestión de eventos basado en un entorno gráfico fácil de usar.

1.3. Metodología

El objetivo de este proyecto es la obtención de un sistema, en formato prueba de concepto, que permita demostrar la capacidad de detección de una posible extracción no autorizada de información por lo que utilizaremos una metodología de gestión de proyectos clásica: la cascada. En los casos donde el cliente no tiene unos conocimientos tecnológicos importantes y donde el proyecto no obtenga una gran implicación por su parte, al no ser un proyecto decisivo para su línea de negocio, es preferible evitar las metodologías ágiles, como Scrum. Si finalmente de esta prueba de concepto deriva en un proyecto de implantación final en sus sistemas evaluaremos la idoneidad de estas nuevas metodologías ya que sería muy probable la aparición de nuevos requisitos (petición de nuevos controles, añadir servidores al proyecto, detección de nuevas situaciones, etc.) según los prototipos se fueran entregando.

La metodología clásica de gestión en proyectos recibe el nombre de cascada ya que la idea se basa en que cada fase requiere de la finalización de sus anteriores, y del producto resultante de éstas, para poder avanzar hasta el siguiente hito del proyecto. Así tenemos una cadena de acciones y tareas que deben realizarse una tras otra, y en un orden específico, para conseguir el objetivo final.

La metodología de proyectos en cascada está basada en ocho fases que van desde la concepción de la necesidad del proyecto (fase 1) hasta la implementación del proyecto (fase 7) y su mantenimiento posterior (fase 8). Cada una de las fases pretende ir acotando de forma realista tanto las necesidades del usuario (fases de concepción, inicialización y análisis) como las propuestas técnicas (fases de diseño, construcción y pruebas) y su posterior puesta en producción y mantenimiento.

En nuestro caso nos encontramos que la empresa solicitante ya ha pasado por las dos fases iniciales del proyecto puesto que ha descubierto una necesidad (detectar la extracción no autorizada de información) y ha evaluado sus capacidades propias para buscar una solución (optando por solicitar asesoría externa)por lo que iniciaremos este proyecto con la recopilación y el análisis de información inicial para posteriormente entrar en las fases de diseño, construcción y pruebas de la prueba de concepto. Ya que el proyecto únicamente consiste en la

demostración de capacidades con una prueba de concepto, y no en una instalación real en un entorno productivo, no consideramos la existencia de la fase de mantenimiento (la última en la metodología en cascada).

1.4. Definición de tareas

Cada fase del proyecto está basada en la realización de diferentes tareas en cada momento con la intención de alcanzar el objetivo final del proyecto y su puesta en marcha en un entorno productivo. Para un correcto avance del proyecto es importante realizar estas tareas de forma completa y consecutiva.

Las dos primeras fases de la metodología en cascada (concepción e iniciación) ya han sido realizadas por la propia empresa, aunque no sea de una forma explícita, ya que ante la detección de un problema (extracción no autorizada de información) se decidió buscar una solución tecnológica y se analizaron diversas opciones por el departamento técnico, optando finalmente por solicitar una consultaría externa debido a la falta de tiempo y experiencia previa de los propios técnicos de la empresa.

TAREA	RESULTADO
Concepción	
-	Las fase de concepción superada antes de iniciar el proyecto
Iniciación	
-	Las fase de iniciación ya superada antes de iniciar el proyecto
Análisis	
Recopilación información	Datos reales de infraestructura informática actual
Análisis tareas informáticos	Capacidades de asumir nuevas tareas
Diseño	
Configuración estándar Snort	Listado de cambios a la configuración estándar para adaptar Snort a las necesidades del proyecto
Estudio reglas existentes	Listado de reglas ya existentes útiles para nuestro caso
Definición nuevas reglas	Listado de eventos que deseamos detectar para los que no existen reglas por lo que deberán desarrollarse
Estudio entornos gráficos	Selección de entornos gráficos
Construcción	
Creación entorno virtual	Servidores virtuales configurados para simular el entorno real
Instalación Snort	Sistema Snort listo para operar sobre el entorno virtual
Desarrollo firmas	Las nuevas firmas definidas anteriormente
Instalación entorno gráfico	Entorno gráfico operativo para la gestión de eventos
Pruebas	
Pruebas firmas	Certificación y ajustes de las nuevas firmas
Pruebas entorno gráfico	Certificación del entorno gráfico
Implementación	
Documentación	Documentos y procedimientos operativos para los técnicos
Memoria y presentación	Documentación definitiva y presentación de la prueba de concepto
Mantenimiento	
-	Al tratarse de una prueba de concepto no hay una fase de mantenimiento

Tabla 1 - Relación de tareas e información obtenida

1.5. Programación temporal

La planificación temporal de este proyecto, como cualquier otro, está basada en una estimación de la carga de trabajo que implican las diferentes tareas extraídas del análisis anterior. Al trabajar con una metodología en cascada, donde cada tarea depende de las previas para su inicio, el resultado de la planificación temporal queda reflejado de forma sencilla en un diagrama de Gantt de tal forma que podemos entender visualmente el porqué del nombre "en cascada".

Siguiendo la metodología en cascada, tras la definición de las tareas a realizar en cada fase del proyecto hemos preparado una planificación temporal base para lo que hemos asignado unas fechas de inicio y una duración estimada a dichas tareas. Aunque podemos mostrar toda esta información de la planificación temporal en formato tabla (cómo la que se puede ver a continuación) el uso de un diagrama de Gantt facilita la visualización de las implicaciones y del formato 'en cascada' de esta metodología.

TAREA	FECHA INICIO	FECHA ENTREGA
Concepción		
-	Realizado previamente. Fuera de planificación.	Realizado previamente. Fuera de planificación.
Iniciación		
-	Realizado previamente. Fuera de planificación.	Realizado previamente. Fuera de planificación.
Análisis		
Recopilación información	14/03/2016	21/03/2016
Análisis tareas informáticos	14/03/2016	21/03/2016
Diseño		
Configuración estándar Snort	22/03/2016	24/03/2016
Estudio reglas existentes	25/03/2016	31/03/2016
Definición nuevas reglas	01/04/2016	14/04/2016
Estudio entornos gráficos	25/03/2016	14/04/2016
Construcción		
Creación entorno virtual	15/04/2016	19/04/2016
Instalación Snort	20/04/2016	20/04/2016
Desarrollo firmas	21/04/2016	27/04/2016
Instalación entorno gráfico	21/04/2016	11/05/2016
Pruebas		
Pruebas firmas	12/05/2016	18/06/2016
Evitar falsos positivos	19/05/2016	01/06/2016
Pruebas entorno gráfico	19/05/2016	23/05/2016
Implementación		
Documentación	02/06/2016	06/06/2016
Memoria y presentación	07/06/2016	13/06/2016
Mantenimiento		
-	Fuera de planificación	Fuera de planificación

Tabla 2 - Planificación temporal

En las siguientes capturas del diagrama de Gantt podemos observar cómo cada fase del proyecto involucra las diferentes tareas, antes indicadas, y su previsión temporal.

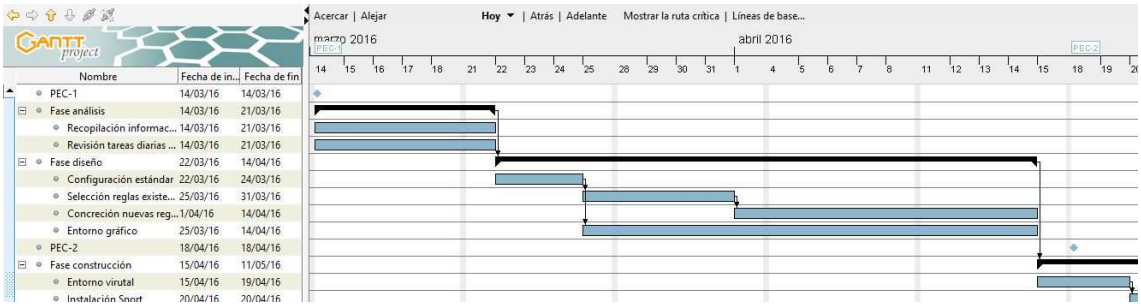

Ilustración 1 - Diagrama de Gantt 1/3 - Fases de análisis y diseño

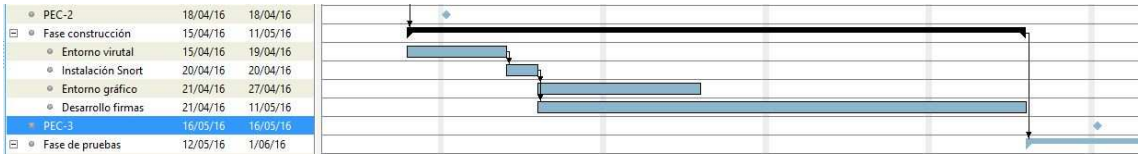

Ilustración 2 - Diagrama de Gantt 2/3 - Fase de construcción

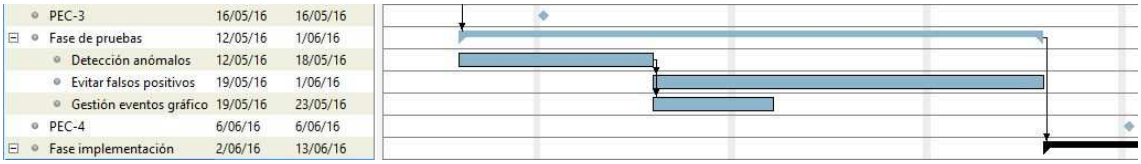

Ilustración 3 - Diagrama de Gantt 3/3 - Fase de pruebas e implementación

Podemos observar claramente cómo el orden de las tareas corresponde a la planificación por fases y que todas tienen como objetivo único la obtención de un producto final que cubra las necesidades del cliente del proyecto: en nuestro caso una prueba de concepto que muestre como una correcta configuración de un sistema Snort podría detectar comportamientos que nos induzcan a pensar en la existencia de una extracción no autorizada de información de la empresa.

2. Diseño de la propuesta

Para diseñar nuestra propuesta nos basamos en unas premisas y requerimientos básicos y en las capacidades reales del sistema Snort definiendo, así, cómo debemos configurar el sistema para conseguir una solución realista y útil para la empresa.

En esta sección del proyecto mostraremos las premisas iniciales sobre las que basamos el desarrollo de la solución que finalmente aplicaremos en la fase de construcción.

2.1. Premisas

En este apartado veremos las premisas sobre las que basaremos la propuesta de prueba de concepto para, posteriormente, definir las situaciones que pretendemos detectar para cumplir el objetivo de activar una alarma en el caso de una posible extracción no autorizada de información.

2.1.1 Infraestructura técnica actual

La empresa ya cuenta con una infraestructura tecnológica que permite la realización de su trabajo diario por lo que nuestra propuesta debe tener en cuenta precisamente qué sistemas están utilizando actualmente como base de su proceso de negocio para poder adaptar nuestro sistema Snort a sus necesidades y capacidades reales.

Tras unos trabajos previos del departamento técnico de la empresa los entornos productivos utilizados son únicamente dos: sistemas basados en Windows 2012 R2 para aquellos servicios que interaccionan con los usuarios (impresión, almacenamiento compartido, ActiveDirectory, DHCP, etc.); y servidores basados en una distribución Linux Ubuntu para los servicios que no tienen un acceso tan directo (servidores web, bases de datos, correo electrónico, etc.).

Con esta infraestructura en mente queda claro que debemos buscar una solución que permita detectar comportamientos anómalos, de los que puedan deducirse un intento de extracción de información no autorizada, en ambos sistemas operativos.

2.1.2 Diferenciación de autorización por horario

El principal problema encontramos en cualquier sistema que intente detectar una extracción no autorizada de información es el hecho de que técnicamente no hay diferencia alguna entre un acceso autorizado y legítimo a un recurso y uno malintencionado. Ya que no es posible diferenciar técnicamente un acceso a información legítimo de uno no autorizado este proyecto asumirá como una extracción de información no autorizada aquellos accesos o conexiones no habituales bien por su propia naturaleza o bien por el horario no habitual de los mismos.

Consideraremos que el acceso a la información de la empresa que se produzca fuera del horario laboral es un indicio de posible extracción no autorizada de información. La definición de horario laboral puede realizarse bien de forma genérica para toda la compañía o bien de forma específica para cada departamento de la empresa. Esta definición horaria es importante para evitar que el sistema Snort acumule falsas alarmas por los accesos a la información ordinarios y legítimos de los empleados. La acumulación de estos falsos positivos derivaría en la saturación de los técnicos que deben revisarlas y la no detección de eventos reales.

Los horarios definidos como laborables para esta empresa se han unificado en las siguientes bandas:

- Departamentos administrativos: L-V 08:00 – 15:00
- Departamentos productivos: L-V 08:00 – 20:00

Por lo tanto, consideraremos que los accesos a la información de carácter administrativo (facturación, nóminas, etc.) realizados desde las 15:00 son susceptibles de ser una extracción no autorizada de información, mientras que para los entornos productivos (datos de almacén, intranets, etc.) el horario de riesgo empieza a las 20:00. Consideraremos los fines de semana como no laborables en cualquier horario.

2.1.3 Accesos no habituales

Además de los indicios de extracción no autorizada de información basados en el horario de aparición de los eventos, en este proyecto también analizaremos aquellos accesos realizados mediante un sistema no habitual. Todos los servidores, sea cual sea su sistema operativo, disponen de un acceso secundario pensado para permitir la gestión de los mismos por parte de los administradores del entorno pero que, por regla general, no son utilizados por los clientes.

Como ejemplos de estos accesos no habituales podríamos indicar la conexión por escritorio remoto al servidor de ficheros corporativo, ya que lo habitual es acceder a él vía recursos compartidos y no abrir una conexión RDP; el acceso como *root*, y no como usuario nominal, a un servidor Linux; o el uso de un usuario de base de datos con permisos totales desde consultas externas al propio servidor. Todas estas acciones pueden ser perfectamente lícitas como parte del trabajo de los administradores de sistemas, pero desde luego no es habitual ni común por lo

que, aunque es previsible la aparición de falsos positivos podemos tener una tasa lo suficientemente baja como para mantener esta monitorización.

2.1.4 Registro de eventos en servidor remoto

La necesidad de este proyecto surge tras un problema previo con uno de los antiguos integrantes del departamento informático de la propia empresa el cual tenía, como es lógico, acceso y permisos completos sobre la plataforma de la propia empresa. Las capacidades técnicas de estos empleados permitirían la eliminación de los eventos y alarmas que sus propias acciones hubieran generado y pudieran incriminarlos, por lo que se solicita que los eventos generados por el sistema Snort se copien en un sistema remoto no gestionado por los propios administradores de sistemas. Con esta característica la empresa cubriría el caso en el que la extracción no autorizada de información sea realizada por uno de los técnicos informáticos, como ya pasó anteriormente.

2.1.5 Entorno de gestión gráfico

Según lo comentado previamente estamos ante una empresa no tecnológica que cuenta con un departamento de explotación de sistemas pequeño y muy atareado y cuya prioridad principal seguirá siendo la resolución de incidencias y puesta en marcha de los nuevos servicios que el resto de departamentos vayan necesitando. Debido a la alta carga de trabajo de los técnicos informáticos presentaremos un entorno de gestión de eventos y alarmas simple y sencillo que permita minimizar el tiempo dedicado a esta nueva tarea de control.

2.2. Características

En el apartado anterior hemos indicado las premisas y condiciones que debía cumplir nuestro proyecto para cubrir las necesidades reales de esta empresa por lo que en los siguientes puntos desarrollaremos las principales características, tanto técnicas como conceptuales, sobre la que trabajaremos para cumplir los requisitos del cliente.

2.2.1 Dos sistemas operativos

Los técnicos de sistemas realizaron un importante trabajo previo de unificación de los entornos informáticos que dan servicio al resto de departamentos de la empresa. Así nos encontramos con dos entornos claramente diferenciados dentro de la empresa, lo cual facilita la creación de un entorno que permita simular el comportamiento de la aplicación Snort en el entorno real.

Para realizar la prueba de concepto creamos dos servidores virtuales, para no interferir en ningún caso con el entorno productivo real, con características que permitan simular el entorno real y donde instalaremos la aplicación Snort:

- Instalaremos un servidor Windows 2012 R2 que simulará ser uno de los servidores de almacenamiento compartido de la empresa (ActiveDirectory, DHCP, carpetas compartidas, etc.)
- Un segundo servidor con sistema operativo Ubuntu y una base de datos MySQL para simular el entorno productivo real.

La instalación de un sistema Snort local en cada uno de los servidores nos permitirá mostrar el correcto funcionamiento del mismo en ambos entornos y cómo lo podemos configurar, en base a las firmas de detección seleccionadas, para cubrir especificidades de cada entorno.

2.2.2 Gestión horaria diferenciada

Debido a la inexistencia de diferencias técnicas entre los accesos a información legítimos y los no autorizados hemos optado por la gestión de algunos eventos en función de la franja horaria de su aparición mientras que otros, debido a los pocos usos legítimos que pueden activarlos, serán analizados en todo momento.

El sistema Snort basa su funcionamiento general en la información almacenada en el fichero *snort.conf*. En dicho fichero encontraremos los datos que necesita el programa para funcionar y las firmas de detección que debe utilizar para monitorizar el entorno. Así pues, la correcta gestión de este archivo es básica para obtener el comportamiento esperado y para que, en función de la franja horaria, se habiliten unas reglas de detección u otras.

Para facilitar la gestión, las reglas de detección se encuentran definidas en unos ficheros externos independientes con terminación *.rules* que son invocados desde el fichero de configuración general. Así para modificar, añadir o quitar alguna firma de detección no hay que hacer cambios en el fichero general sino únicamente en aquel fichero que contiene las reglas que queremos gestionar. La instalación básica de Snort incluye una serie de archivos de firmas de detección organizados por el tipo de ataque o el protocolo utilizado: así en el fichero *smtp.rules* encontraremos reglas que hacen referencia al protocolo en envío de correos electrónicos; mientras que en fichero *scan.rules* estarán ubicadas aquellas firmas que intentan detectar un escaneo a nuestros servidores.

Para añadir nuevas reglas a la monitorización tan sólo tenemos que modificar el fichero .rules existente que mejor se adapte a la situación o bien crear un nuevo fichero *misReglas.rules* que sea invocado desde el fichero de configuración general *snort.conf*.

Para este proyecto, debido a que queremos activar y desactivar diferentes firmas de detección en función del horario vamos a crear diferentes ficheros de firmas que invocaremos, o no, en función del horario de ejecución de Snort. De esta forma crearemos estos ficheros:

- *revisarSiempre.rules*:
  En este fichero incluiremos aquellas firmas que buscan detectar comportamientos y accesos poco habituales que por su escaso uso y potencial riesgo hemos decido que estén activas siempre. Para ello en el fichero *snort.conf* habrá siempre una invocación a este fichero (*include revisarSiempre.rules*) permitiendo ser utilizadas por el programa en cualquier momento.
- *revisarAdministrativos.rules*:
  En este fichero incorporaremos aquellas firmas que usaríamos para detectar comportamientos anómalos, por el horario, en el entorno administrativo de la empresa. Así la invocación de este archivo (*include revisarAdministrativos.rules*) se realiza únicamente en el horario no laboral previamente definido: tardes, noches y fines de semana.
  Para esta prueba de concepto hemos definido como entorno administrativo el servidor de archivos Windows 2012.
- *revisarProductivos.rules*:
  En este fichero, de forma similar al anterior, encontraremos las firmas de detección de usos no correctos para el entorno productivo que, en esta prueba de concepto, hemos definido como el servidor Ubuntu con base de datos MySQL. De igual forma que con el

fichero anterior cuando estemos en horario no laborar se incluirá las reglas con la llamada correspondientes (*include revisarProductivos.rules*).

Una vez definido que mantendremos tres ficheros de reglas independientes y que serán utilizados en función, ya no del tipo de tráfico a analizar, sino de la franja horaria en la que lo analizamos, hay que definir otros tantos ficheros de configuración *snort.conf* para que sean los invocados en el arranque de la aplicación indicando qué ficheros de firmas que queramos aplicar:

- *snortLaborable.conf*
  Este sería el fichero el utilizado durante el horario laboral común (L–V 08:00 – 15:00) y únicamente incluiría la invocación al fichero *revisarSiempre.rules*.
- *snortAdministrativo.conf*
  Este sería el fichero de configuración que utilizaría el programa para aquellas franjas horarias donde el departamento administrativo está cerrado, pero no así el productivo (L–V 15:00 – 20:00). En este fichero de configuración invocaríamos dos ficheros de firmas: *revisarSiempre.rules* y *revisarAdministrativos.rules*)
- *snortTodos.conf*
  Este fichero de configuración sería el utilizado en las franjas horarias donde no debería haber nadie trabajando (L–V 20:00 – 08:00 y fines de semana) e incluiría los tres ficheros de firmas: *revisarSiempre.rules*, *revisarAdministrativos.rules* y *revisarProductivos.rules*.

Con este montaje obtenemos tres configuraciones diferentes que modifican el comportamiento del sistema en función de la franja horaria de tal forma que podemos activar las firmas de detección que nos interesan para detectar comportamientos anómalos.

Una vez definidos los diferentes archivos de configuración para que el sistema se adapte a la franja horaria correspondiente y aplique, o no, las firmas de detección hay que preparar una planificación horaria para que el propio sistema operativo gestione el arranque de Snort con la configuración correspondiente. Para realizar esta gestión horaria utilizaremos dos utilidades incorporadas en los propios sistemas operativos: *crontab* en Ubuntu y *Programador de Tareas* en entornos Windows 2012. La planificación horaria sería la siguiente:

- L – V 08:00: Parar la ejecución actual de Snort y arrancar de nuevo usando el fichero snortLaborable.conf
- L – V 15:15: Parar la ejecución actual de Snort y arrancar de nuevo usando el fichero snortAdministrativo.conf
- L – V 20:15: Parar la ejecución actual de Snort y arrancar de nuevo usando el fichero snortTodos.conf

Cabe indicar que el margen de 15 minutos tras el cierre de los diferentes departamentos es debido a que, por experiencia previa sabemos que siempre hay alguna persona que está terminando una tarea y que no sale a su hora en punto. Si lo deseamos podríamos cambiar esta ejecución a la hora exacta pero muy probablemente nos encontraríamos en un entorno con falsos positivos casi a diario por lo que se reduciría su efectividad.

2.2.3 Gestión de eventos por Syslog

Una de las premisas de este proyecto es que, aunque el usuario malintencionado que pretende extraer información disponga delos conocimientos técnicos necesarios, no se pueda falsear la información que revisan los técnicos de sistemas. Para evitar el posible borrado de eventos y

alarmas enviaremos una copia a un servidor Syslog remoto que no sea controlado por los técnicos de la empresa. Con esta configuración conseguimos tener una copia no adulterada que no puede ser modificada por ningún empleado de la corporación.

Para poder realizar este montaje deberemos modificar el fichero de configuración *snort.conf* (en realidad los tres ficheros definidos anteriormente) para incluir una línea donde especificaremos cuál es el servidor de syslog remoto al que debe enviar la información y qué nivel de información deseamos enviar (nivel de criticidad, contenido de las trazas, etc.).

Dentro del sector de configuración de la salida de información de Snort habilitaremos la una línea de configuración siguiendo el siguiente esquema:

output alert_syslog: host=host:port, LOG_AUTH LOG_ALERT

2.2.4 Gestión gráfica de eventos

Con el objetivo de minimizar el tiempo de gestión de los eventos por parte de los técnicos de la empresa ofreceremos un entorno gráfico simple y sencillo. Al no tener ningún requerimiento de integración con herramientas de gestión ya existentes en la empresa buscaremos un entorno propio para este proyecto que permita de forma rápida detectar si ha surgido algún evento digno de ser analizado.

Snort no tiene ningún entorno gráfico predefinido y estándar, pero gracias a su licencia de código abierto se han desarrollado diferentes sistemas gráficos que buscan facilitar la gestión diaria de los responsables técnicos. Presentaremos, pues, una solución que permita revisar las alarmas de forma simple y rápida.

3. Solución técnica

La solución técnica propuesta se basa en la demostración de la detección de eventos de un sistema Snort tanto en servidores Windows como Linux. Aunque no suele ser la configuración habitual en los entornos productivos reales, debido al gran número de servidores a controlar, para la prueba de concepto hemos optado por la instalación de Snort en local en cada uno de los equipos mostrando así la adaptación de Snort a cualquiera de los entornos.

En los siguientes puntos desarrollaremos en profundidad las diferentes configuraciones, tanto de Snort como de los propios servidores, así como la creación y adaptación de las diferentes firmas de detección.

En este proyecto no incluimos la descripción de los procesos de instalación básica de ninguno de los sistemas involucrados (sistemas operativos, Snort, herramientas complementarias, etc.) ya que los consideramos fuera del alcance de esta prueba de concepto.

3.1 Descripción laboratorio

Para esta prueba de concepto hemos montado un laboratorio usando las posibilidades de crear entornos virtuales que nos ofrece VirtualBox (VirtualBox, 2016).

Para poder realizar esta prueba de concepto hemos creado y configurado dos máquinas virtuales para simular el entorno productivo de la empresa:

- Servidor Windows 2012 R2 con dos carpetas compartidas para simular la operativa de acceso a los documentos corporativos.

- Servidor Ubuntu 16.04LTS con una instalación básica de MySQL para simular la misma configuración de la empresa.

Sobre estos dos sistemas operativos hemos realizado la instalación y configuración de Snort, y demás herramientas auxiliares, para poder mostrar las posibilidades técnicas de detección de extracción no autorizada de información.

3.2 Configuración Snort

Existe una amplia documentación oficial (Snort.org, 2016), y en múltiples páginas webs no oficiales, sobre la configuración de un sistema Snort por lo que en este proyecto tan sólo recalcaremos aquellas opciones que más aplicación han tenido en nuestra prueba de concepto. Así, en este proyecto no documentaremos ni comentaremos las opciones que no tengan una aplicación específica para nuestra prueba de concepto.

El funcionamiento del sistema Snort se basa en la información de configuración que encontramos en el fichero *snort.conf*. La ubicación predeterminada de dicho fichero es en */etc/Snort* para los sistemas Linux y en *C:\Snort\etc* para los entornos Windows. En dicho fichero podemos encontrar toda la configuración que indica a Snort qué listado de firmas utilizar, cómo generar los eventos (formato, ubicación, etc.), etc. por lo que será el punto central de modificaciones para nuestro proyecto.

3.2.1 Syslog remoto

Debido a los acontecimientos previos que desembocaron en la solicitud de este proyecto para la detección de extracción no autorizada de información, desde la primera reunión y toma de contacto se hizo patente la necesidad de mantener una copia de los eventos generados por el sistema en un servidor remoto no gestionado directamente por los técnicos informáticos.

Para activar esta función hay que modificar el fichero *snort.conf* en la sección *Output Configuration* para que, además de registrar los eventos en los diferentes ficheros habituales, lance una copia en el servidor Syslog remoto. En el manual de Snort podemos ver todas las configuraciones de registros de salida posibles, si bien en nuestro caso la única instrucción a añadir queda como sigue:

```
output alert_syslog: server=10.80.80.80:514 LOG_AUTH LOG_ALERT
```

Con esta instrucción indicamos al sistema Snort que deseamos activar una copia de la salida hacía un servidor syslog remoto (en nuestro caso 10.80.80.80 en puerto 514) enviando la información de eventos con prioridad “alerta”.

En el caso de servidores Linux, como es el caso de esta prueba de concepto, además de configurar correctamente el sistema Snort hay que autorizar el envío de la información desde el propio sistema operativo. Se trata de una medida de seguridad del propio sistema operativo, ya que la aplicación utiliza los servicios del sistema operativo para el envío de eventos syslog a servidores remotos. En el caso de nuestro servidor Ubuntu es suficiente con modificar el fichero */etc/rsyslog.d/50-default.conf* añadiendo una línea que permita enviar los eventos de prioridad alerta al servidor 10.80.80.80.

Para la realización de la prueba de concepto configuramos el servidor Linux para que enviase la información al servidor Windows 2012 R2 donde habíamos instalado un servidor de Syslog (Kiwi Syslog Server (SolarWinds, 2016)) donde quedase registrados los eventos. Para la prueba de concepto hemos utilizado la versión de evaluación de este software por ser fácil de utilizar y

configurar si bien en caso de aplicarlo a un entorno productivo real habría que adaptar esta solución a las necesidades reales.

En la siguiente ilustración podemos observar una captura de pantalla donde podemos ver que el servidor Linux ha detectado un evento "UOC – Conexión SSH" y que el mismo queda reflejado en los eventos que registra el servidor Kiwi Syslog Server.

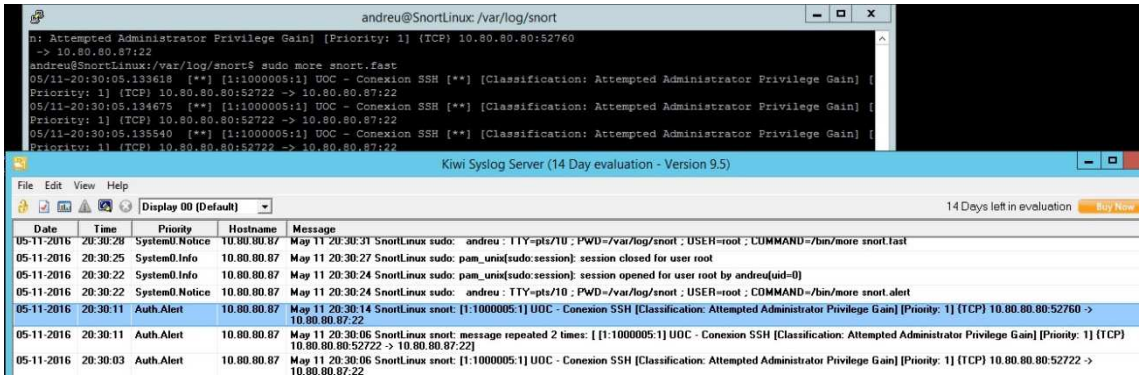

Ilustración 4 - Ejemplo de evento capturado en servidor Syslog remoto

3.2.2 Definición variables

En un entorno productivo habitual se suele disponer de diferentes grupos de servidores para realizar distintas funciones (servicios web, correo, FTP, etc.). El sistema Snort incorpora una gran cantidad de firmas de detección que buscan detectar eventos maliciosos en función de los protocolos de cada servicio. Debido a la cantidad de reglas para servicios totalmente diferentes que puede gestionar un sistema IDS, Snort permite la creación de diferentes agrupaciones de servidores para así poder realizar únicamente los análisis necesarios al tráfico: revisar las firmas del protocolo de envío de correo (SMTP) en el tráfico que reciben los servidores web (que debería ser HTTP, HTTPS) puede ser una sobrecarga inútil al sistema. Así pues, aunque esta característica no sea de uso obligatorio, es conveniente su uso, siempre que sea posible, para optimizar el funcionamiento del sistema.

Para nuestra prueba de concepto definiremos una agrupación de servidores en base a su dirección IP usando la definición de variables *ipvar*. El la documentación de Snort podremos encontrar toda la información sobre la estructura y definición de estas variables (Snort, 2016).

Utilizaremos este tipo de variable en la regla propuesta para la detección de tráfico entre los frontales web y la base de datos MySQL en horario no laboral.

3.2.3 Definición de umbrales de alarma

Una de las características más útiles en la definición de firmas de detección es la posibilidad de definir umbrales de activación. La definición y configuración de estos umbrales está definida de forma extensa en la documentación oficial (Snort, 2016), pero, para facilitar el entendimiento del porqué de su uso, haremos una pequeño resumen a continuación de los tres tipos existentes:

- Por su primera aparición en un tiempo definido. Si utilizamos la opción *limit* la alerta generará un evento por la primera activación de la regla e ignorará los siguientes eventos hasta que haya pasado el intervalo de tiempo definido. Este tipo de umbral puede ser muy útil para detectar un evento que posteriormente sea repetitivo y donde no sea útil llenar el registro con eventos duplicados (control de acceso a puertos específicos).

- Por alcanzar un número mínimo de repeticiones. Utilizando la opción *threshold* conseguiremos que generar una alerta cuando se alcancen el número de repeticiones indicado en el tiempo definido. Si hay menos eventos del umbral definido no se activa la alerta. Este umbral es práctico si tan sólo deseamos detectar una repetición excesiva de un evento (ataques DoS, DDoS, etc.).
- Por una combinación de ambos. Usando la opción *both* podemos generar una alarma en la primera aparición del evento si, posteriormente, alcanzamos el número de repeticiones en el período de tiempo definido en el umbral.

Las reglas de acceso a servidores que definimos para esta prueba de concepto están basadas en las conexiones a puertos específicos por lo que en una comunicación normal generan una gran cantidad de eventos repetitivos que podrían llevar a que descartemos otras alarmas. Debido a esto utilizaremos la opción *limit* en nuestras firmas para minimizar el ruido.

3.3 Definición de reglas de detección

La gran potencia del sistema IDS Snort, y probablemente la principal causa de su éxito, está en la diversidad y adaptabilidad del sistema de creación de reglas de detección de situaciones. La posibilidad de crear nuevas firmas específicas para cada necesidad, y el uso de una licencia que lo permita, ha creado una comunidad de usuarios que desarrollan y comparten nuevas reglas generando un sistema cada vez más capaz. En el manual oficial de Snort (Snort, 2016) podemos encontrar, de forma totalmente abierta, toda la información para crear nuevas firmas o adaptar las ya existentes a nuestras necesidades específicas.

En este apartado definiremos y mostraremos las reglas creadas para la detección de los eventos que hemos considerado como ejemplos de las capacidades del sistema Snort para detectar comportamientos anómalos que pudieran estar relacionados con la extracción no autorizada de información.

Cabe indicar que todas las firmas de detección incorporan un identificador unívoco que permite su localización y gestión de forma cómoda y sencilla. Con estos identificadores únicos se consigue una total claridad y se evitan errores en la comunidad. Ya que cada usuario puede crear sus propias reglas, que también deben llevar un identificador, se han definido dos bloques de numeración para separar las reglas públicas y distribuidas de las reglas propias (Barjatiya, 2016): los números inferiores al millón (1.000.000) están reservados y sólo podrán asignarse por los responsables de distribución de Snort mientras que la numeración superior a esa cifra es de libre uso para los clientes. Así todas las reglas creadas por nosotros, y que sólo vayan a utilizarse en nuestros equipos propios, deben tener un identificador superior a $1_1$000.000.

Presentaremos la explicación de las diferentes reglas utilizadas en esta prueba de concepto en función de la franja horaria en las que estarán activas.

3.3.1 Reglas a revisar en todo momento

Aunque, como ya hemos comentado previamente, una de las premisas de este proyecto es la detección de eventos sospechosos por la franja horaria de su aparición, existen una serie de eventos que, por su excepcionalidad, podemos considerar posibles precursores de una extracción no autorizada de información sea cual sea el horario de su aparición. En este apartado desgranaremos las firmas creadas para detectarlos y el motivo del mismo.

3.3.1.1 Detección de conexiones por Escritorio Remoto

La empresa objeto de este proyecto disponed e diferentes servidores, basados en Windows 2012 R2, que realizan funciones típicas de *File&Print* por lo que el uso habitual por parte de los usuarios es el acceso a los documentos compartidos en red. Con esta premisa en mente vemos que, en general, ningún usuario tiene necesidad de conectarse a estos servidores mediante el servicio de Escritorio Remoto por lo consideramos que la aparición de este tipo de conexiones un indicio que debe generar un evento a revisar. El servicio de Escritorio Remoto es útil para los administradores de sistemas por lo que es común mantenerlo activo para facilitar su administración.

Con la siguiente regla detectaríamos la conexión al puerto 3389 (el que utiliza el servidor RDP por defecto) a uno de nuestros servidores Windows (el 10.80.80.80 en nuestro ejemplo). Ya que las conexiones RDP generan una gran cantidad de tráfico en el puerto 3389 hemos definido unos umbrales para que genere una alarma cada dos minutos.

```
alert tcp any any -> 10.80.80.80 3389 (msg:"UOC - Conexion RDP"; threshold:type limit,
track by_src, count 1, seconds 120; sid:1000002; rev:1;)
```

Para entorno productivos sustituiríamos la dirección IP 10.80.80.80 del ejemplo por una variable $RDP_SERVERS donde haya el listado de direcciones IP de los servidores Windows con RDP activo. De esta forma aumentaríamos la eficiencia del servicio Snort ya que evitaríamos que el sistema analizase esta firma contra los servidores Linux donde, simplemente, este servicio no está operativo.

Así pues, esta regla generaría una entrada en el Visor de Eventos de un sistema Windows como podemos ver en la siguiente captura de pantalla:

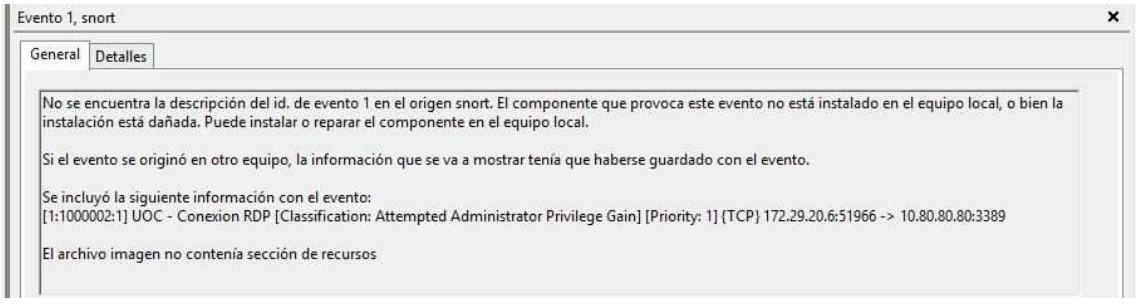

Ilustración 5 - Evento de conexión RDP en el Visor de eventos de Windows

3.3.1.2 Detección de conexiones por ssh

Además de los servidores Windows 2012 R2, la empresa dispone de varios servidores Linux, basados en Ubuntu, que son utilizados por los usuarios a través de aplicaciones (páginas web, accesos SQL, etc.) por lo que la mera conexión por ssh, que al igual que ocurre con las conexiones RDP en los entornos Windows son necesarias para la correcta gestión de la plataforma, al servidor sería un indicio de un comportamiento anómalo.

Con la siguiente regla detectaríamos la conexión al puerto 22 (el que utiliza el servidor ssh por defecto) a uno de nuestros servidores Ubuntu (el 10.80.80.87 en nuestro ejemplo). Ya que las conexiones ssh generan una gran cantidad de tráfico hemos definido unos umbrales para que genere una alarma cada dos minutos.

```
alert tcp any any -> 10.80.80.87 22 (msg:"UOC - Conexion SSH"; threshold:type limit,
track by_src, count 1, seconds 120; sid:1000001; rev:1;)
```

En la captura inferior podemos ver la importancia de definir unos umbrales correctos en las firmas de detección. Cuando realizamos esta captura no habíamos establecido umbral alguno por lo que se generaron tantos eventos como paquetes se recibían en el puerto 22, el usado por el servicio ssh, y tal como vemos hay múltiples alarmas en el mismo segundo (23:47:12 del 10 de mayo) con lo que se llenaría cualquier registro de eventos, dificultando la gestión de eventos y facilitando que perdiésemos información.

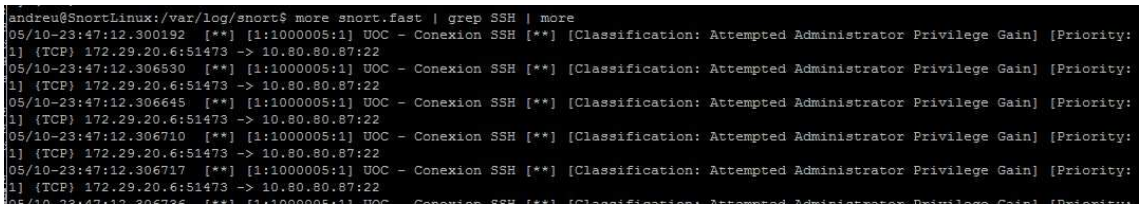

Ilustración 6 - Captura del evento de conexión por ssh en el fichero de eventos snort.fast

3.3.1.3 Detección de conexiones MySQL con usuario root

Las bases de datos, al igual que los sistemas operativos, disponen de un usuario que tiene accesos completos a todo el entorno de la base de datos y que puede realizar cualquier acción sobre la estructura y datos almacenados. Este usuario es imprescindible para el correcto funcionamiento del sistema, por lo que no puede ser eliminado ni deshabilitado, pero debido a sus capacidades es necesario mantener una política de seguridad básica consistente en crear nuevos usuarios en la base de datos que tengan únicamente los permisos necesarios para su función (lectura o escritura en tablas específicas, etc.). En un entorno corporativo ninguna aplicación debería utilizar el usuario administrador, denominado *root* en entornos MySQL, por lo que la detección del uso del mismo es un evento destacable y aviso de un intento de conexión que puede derivar en una fuga de información.

En este caso nos encontramos que ya existe una firma de detección dentro del sistema Snort que detecta las conexiones por lo que no hace falta generar una nueva. Podemos ver que el identificador (sid:1775) pertenece al rango reservado para firmas distribuidas de forma oficial (Caswell & Houghton, 2016). Como la autenticación se realiza únicamente una vez por cada conexión no es necesario utilizar ningún tipo de umbral en la definición ya que no habrá repetición de eventos.

> alert tcp any any -> 10.80.80.87 3306 (msg:"MYSQL root login attempt"; content:"root|00|"; classtype:protocol-command-decode; sid:1775; rev:2;)

Para poder comprobar el correcto funcionamiento de la firma hemos instalado el software HeidiSQL (Becker, 2016) en nuestro servidor Windows. Configuramos la aplicación para que intente acceder al servicio MySQL que se encuentra operativo en el servidor Ubuntu. Como vemos en la captura inferior generamos un evento en el sistema indicando el intento de conexión.

En la misma captura podemos ver que la conexión da un error de denegación de acceso ya que, por defecto, MySQL está configurado para no aceptar conexiones del usuario root desde fuera del propio servidor donde está instalado. Se trata de una política de seguridad predefinida en la configuración básica de la propia base de datos que, aunque podríamos deshabilitar, es preferible mantener.

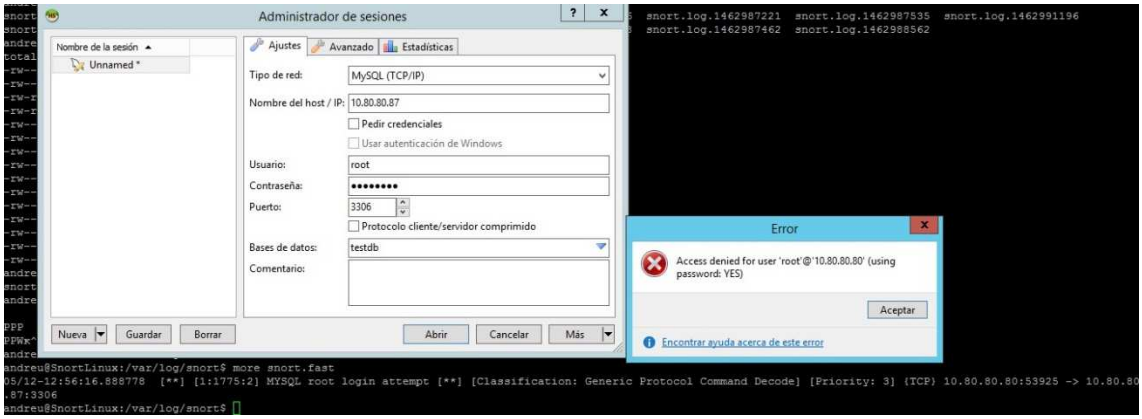

Ilustración 7 - Generación y captura del evento de intento de acceso con usuario root

3.3.1.4 Tráfico FTP saliente

Una de las formas de extracción de información no autorizada es, una vez hayan accedido al servidor deseado, enviar los archivos deseados a un servidor FTP externo. La idea es generar una alarma cuando detectemos conexiones FTP con origen el servidor analizado.

Con la siguiente regla buscaremos conexiones con origen nuestro servidor de ejemplo (10.80.80.80) contra cualquier servidor externo en el puerto 21 (el estándar de FTP). Nuevamente definiremos un umbral para evitar la aparición de múltiples alarmas. La regla quedaría como sigue:

> alert tcp 10.80.80.80 any -> any 21 (msg:"UOC - FTP saliente"; threshold:type limit, track by_dst, count 1, seconds 120; sid:1000005; rev:1;)

Esta firma de detección, al igual que las siguientes de control de HTTP y HTTPS salientes, hemos configurado para generar un evento independiente por cada servidor externo al que se quiera conectar (*track by_dst*). De esta forma podremos tener un registro de a qué servidores ha intentado conectarse.

En la captura inferior podemos observar cómo queda registrado el evento en el Visor de sucesos de Windows.

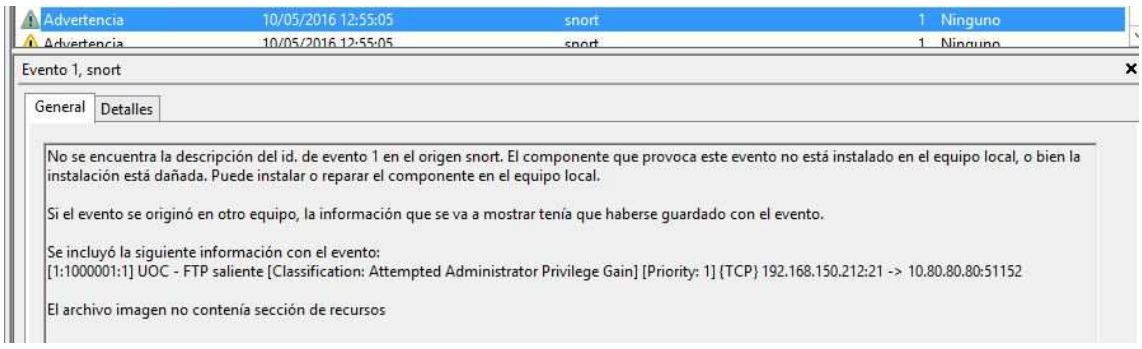

Ilustración 8 - Evento de una conexión FTP el Visor de Eventos de Windows

3.3.1.5 Tráfico HTTP saliente

Aunque las conexiones FTP son más eficientes para el envío de archivos son de uso menos común por lo que es probable que estén más restringidas a nivel de firewall corporativo. Debido a esto es posible que un usuario malintencionado intentase utilizar alguna herramienta web para extraer la información (sistemas almacenamiento en la nube tipo Dropbox, servidores de

correo web, servicios de hosting, etc.) por lo que hemos desarrollado una firma que detecta las conexiones al puerto 80 con origen los servidores monitorizados. Al igual que en los casos anteriores hemos definido un umbral para evitar la repetición de eventos.

> alert tcp 10.80.80.80 any -> any 80 (msg:"UOC - HTTP saliente”; threshold:type limit, track by_dst, count 1, seconds 120; sid:1000003; rev:1;)

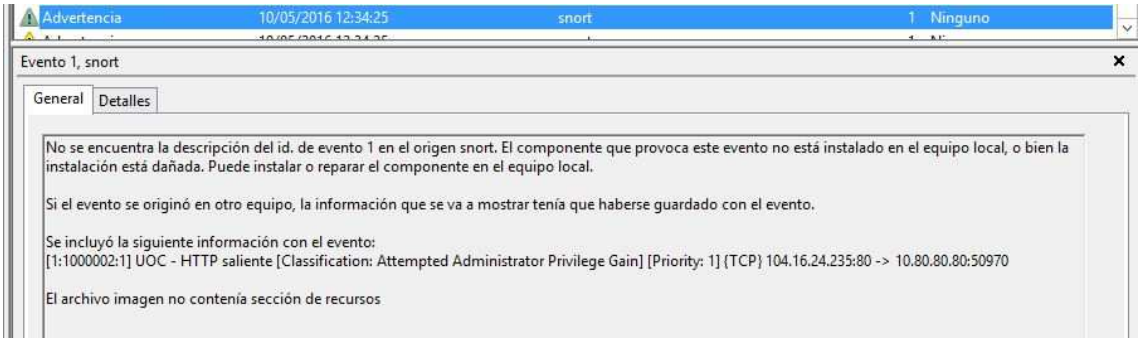

Ilustración 9 - Evento de conexión HTTP en el Visor de Eventos de Windows

3.3.1.6 Tráfico HTTPS saliente

En esta regla hemos seguido el mismo concepto y funcionamiento que en las dos anteriores, pero analizando el puerto 443 que es el corresponde al servicio HTTPS.

> alert tcp 10.80.80.80 any -> any 443 (msg:"UOC - HTTPS saliente”; threshold:type limit, track by_dst, count 1, seconds 120; sid:1000004; rev:1;)

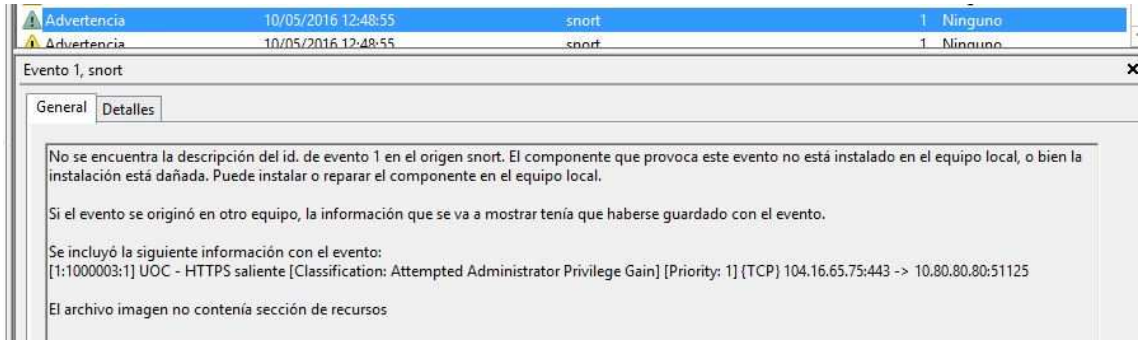

Ilustración 10 - Evento de conexión HTTPS en el Visor de Eventos de Windows

3.3.2 Reglas departamento administración

Con estas reglas pretendemos detectar el acceso a los recursos compartidos utilizados por el departamento administrativo de la empresa fuera del horario laboral habitual.

Para realizar estas firmas hemos capturado el tráfico existente entre un equipo cliente Windows y las carpetas compartidas en el servidor Windows 2012 R2. Para estudiar el tráfico existente hemos utilizado la herramienta gratuita Wireshark (Wireshark, 2016) de tal forma que hemos podido analizar el contenido de los paquetes. Este análisis nos muestra cómo, usando el protocolo SMB2, se solicita el acceso a los recursos compartidos. Una vez conocida la forma cómo este protocolo codifica las peticiones hemos generado sendas firmas de detección que generen alarma cuando se accede a los dos directorios compartidos definidos previamente (RecursosHumanos y Facturacion).

En la captura inferior podemos ver por ejemplo como se estructura la petición para acceder a la carpeta \\10.80.80.80\RecursosHumanos.

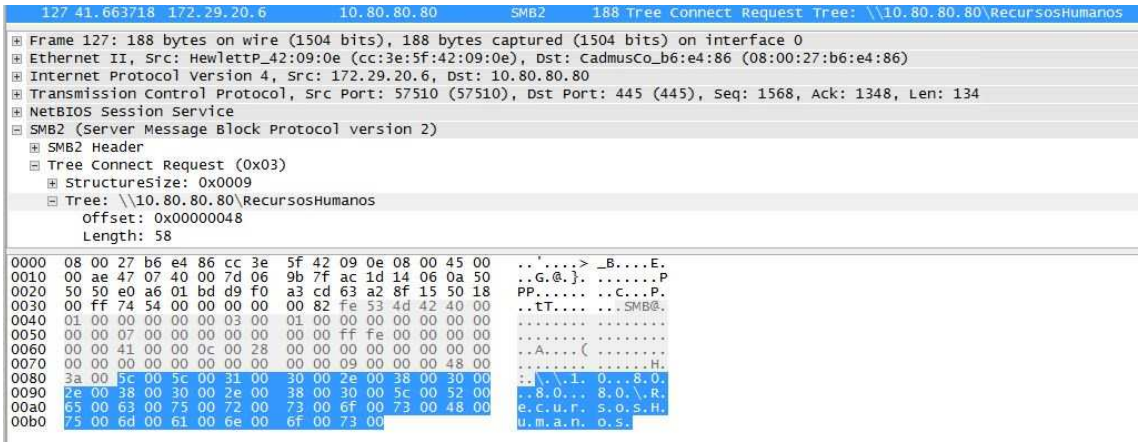

Ilustración 11- Captura de tráfico para analizar las peticiones a recursos compartidos

3.3.2.1 Acceso a directorio Recursos Humanos

Basándonos en las capturas obtenidas con Wireshark podemos detectar que el nombre del directorio accedido aparece en la petición SMB2 de tal forma que entre cada uno de los caracteres del nombre se utiliza el separador 00 hexadecimal por lo que en nuestra firma buscaremos la cadena correspondiente al directorio. Al igual que en casos anteriores se añade un umbral para evitar la aparición de múltiples alarmas.

> alert tcp any any -> 10.80.80.80 445 (msg:"UOC - Acceso directorio Recursos Humanos"; content:"R|00|e|00|c|00|u|00|r|00|s|00|o|00|s|00|H|00|u|00|m|00|a|00|n|00|o|00|s"; threshold:type limit, track by_src, count 1, seconds 120; sid:1000012; rev:1;)

En la siguiente captura podemos ver cómo la firma genera un evento al acceder al directorio compartido desde un equipo cliente:

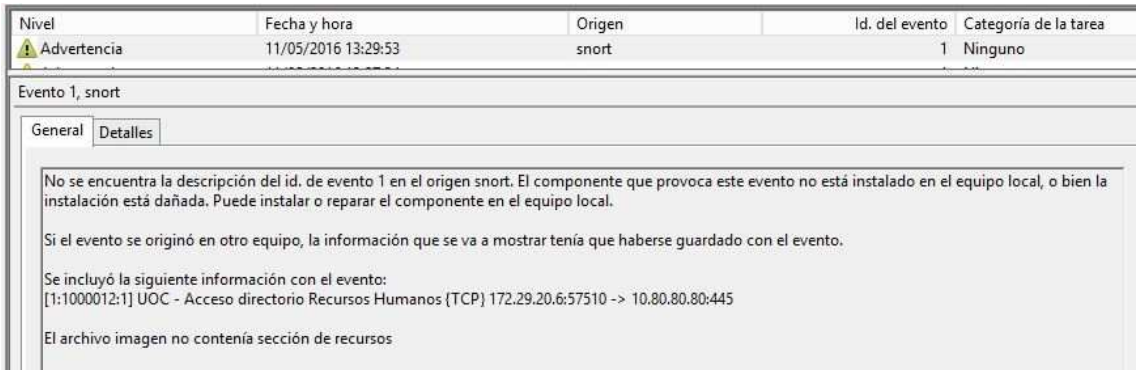

Ilustración 12 - Evento de acceso al directorio compartido RecursosHumanos

3.3.2.2 Acceso a directorio Facturacion

Para esta nueva firma nos basaremos exactamente en el mismo principio cambiando únicamente la cadena de texto a buscar en los paquetes del protocolo SMB2 para localizar la cadena “Facturacion” dentro del paquete de petición SMB2:

> alert tcp any any -> 10.80.80.80 445 (msg:"UOC - Acceso directorio Facturacion"; content:"F|00|a|00|c|00|t|00|u|00|r|00|a|00|c|00|i|00|o|00|n"; threshold:type limit, track by_src, count 1, seconds 120; sid:1000013; rev:1;)

En la captura inferior vemos, al igual que en el caso anterior, el evento generado dentro del Visor de Eventos del servidor Windows donde esté Snort operativo.

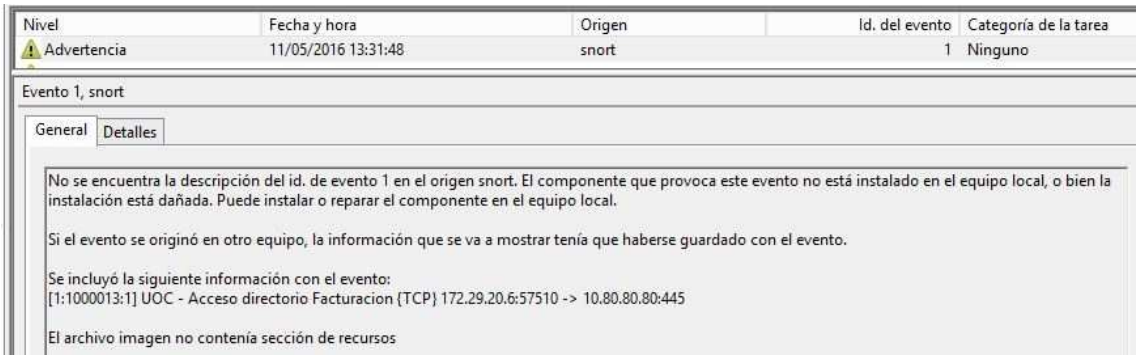

Ilustración 13 - Evento de acceso al directorio compartido Facturacion

3.3.3 Reglas departamento productivo

En este apartado explicaremos la regla de detección creada para detectar el uso de una aplicación web del departamento productivo fuera del horario habitual. Para optimizar al máximo el funcionamiento de Snort, y como ejemplo de uso de las variables antes explicadas, hemos definido una variable Snort de tal forma que esta firma sólo se analice cuando haya una conexión al servidor MySQL desde uno de los frontales web.

Primero definiremos cuáles son los dos frontales web, que tienen las direcciones IP 172.29.20.5 y 172.29.20.6, con la siguiente línea en el fichero de configuración de Snort:

```
ipvar FRONTALES_WEB [172.29.20.5,172.29.20.6]
```

Una vez definida la variable podemos utilizarla en tantas firmas de detección como deseemos haciendo referencia a ella con el código $FRONTALES_WEB.

3.3.3.1 Acceso MySQL desde Frontales

Para detectar el uso fuera del horario habitual generaremos un evento cuando haya una conexión al servidor MySQL desde uno de los frontales web definidos. La regla sería la siguiente:

```
alert tcp $FRONTALES_WEB any -> 10.80.80.87 3306 (msg:"UOC - Acceso MySQL";
sid:1000006; threshold:type limit, track by_src, count 1, seconds 120; rev:1;)
```

```
andreu@SnortLinux:/var/log/snort$ sudo more snort.fast | grep MySQL
05/12-10:01:49.356744  [**] [1:1000006:1] UOC - Acceso MySQL [**] [Priority: 0] {TCP} 172.29.20.6:54741 -> 10.80.80.87:3306
05/12-10:01:49.894686  [**] [1:1000006:1] UOC - Acceso MySQL [**] [Priority: 0] {TCP} 172.29.20.6:54741 -> 10.80.80.87:3306
05/12-10:01:50.430934  [**] [1:1000006:1] UOC - Acceso MySQL [**] [Priority: 0] {TCP} 172.29.20.6:54741 -> 10.80.80.87:3306
05/12-10:02:36.772086  [**] [1:1000006:1] UOC - Acceso MySQL [**] [Priority: 0] {TCP} 172.29.20.6:54745 -> 10.80.80.87:3306
05/12-10:02:37.309031  [**] [1:1000006:1] UOC - Acceso MySQL [**] [Priority: 0] {TCP} 172.29.20.6:54745 -> 10.80.80.87:3306
05/12-10:02:37.846784  [**] [1:1000006:1] UOC - Acceso MySQL [**] [Priority: 0] {TCP} 172.29.20.6:54745 -> 10.80.80.87:3306
andreu@SnortLinux:/var/log/snort$
```

Ilustración 14 - Eventos de acceso al servidor MySQL desde frontales web

3.4 Gestión horaria

Tal como hemos comentado previamente las reglas de detección de Snort no permiten diferenciar su comportamiento en función de la hora de aparición por lo que nuestra estrategia consistirá en forzar la parada y arranque del software Snort cargando la configuración adecuada para la franja horaria deseada.

Aunque estamos ante una prueba de concepto con una instalación de Snort en local en los propios servidores y estas configuraciones pueden no ser iguales en una instalación en red en

un entorno productivo, indicaremos los métodos utilizados para activar las configuraciones adecuadas en cada entorno.

En nuestra prueba de concepto el servidor Linux gestionaba la base de datos MySQL del departamento productivo por lo que el comportamiento debe cambiar en función de su horario laboral (L-V 08h-20h). Para poder hacerlo configuraremos dos entradas de crontab (Toro, 2016) para que cada día, de lunes a viernes, a las 08:00 se cargue la configuración que gestiona únicamente las firmas de detección del fichero *revisarSiempre.rules* y a las 20:15 cargue la configuración que añade las firmas del fichero *revisarProductivo.rules*. La configuración quedaría como sigue:

```
andreu@SnortLinux:~$ sudo crontab -l
0 8 * * 1-5 pkill snort; snort -i eth0 -l /var/log/snort -c /etc/snort/snortLaborable.conf
15 20 * * 1-5 pkill snort; snort -i eth0 -l /var/log/snort -c
/etc/snort/snortProductivo.conf
andreu@SnortLinux:~$
```

Aunque la explicación del funcionamiento del sistema crontab queda fuera del alcance de este proyecto podemos explicar usando la primera línea donde indicamos al servidor que en el minuto indicado (0), de la hora elegida (8), sea cual sea el día del mes (*), sea cual sea el mes (*), de lunes a viernes (1-5) ejecute el comando indicado. Así de lunes a viernes, sin importar la fecha, a las 8:00 se ejecutará el comando *"pkill snort; snort -i eth0 -l /var/log/snort -c /etc/snort/snortLaborable.conf"*.

Para los servidores Windows utilizamos la herramienta Programador de Tareas para que lance Snort con la configuración adecuada. Podemos ver un ejemplo en la siguiente captura de pantalla.

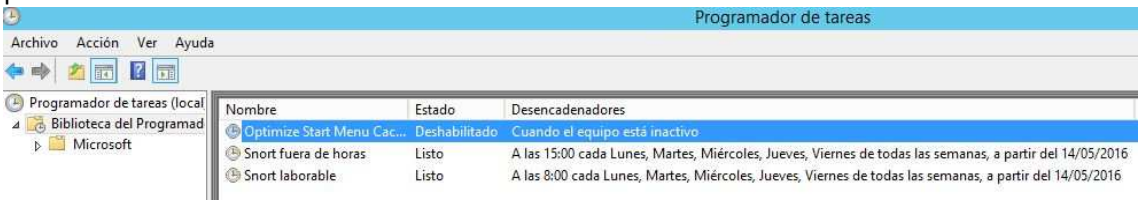

Ilustración 15 - Captura del Programador de Tareas configurado

Aunque para nuestra prueba de concepto hemos realizado dos instalaciones independientes de Snort, una en cada servidor de ejemplo, podríamos resumir la planificación horaria de nuestra propuesta en la siguiente tabla:

	Lunes – Viernes	**Sábado - Domingo**
00:00 – 08:00	snortTodos.conf	snortTodos.conf
08:00 – 15:15	SnortLaborable.conf	
15:15 – 20:15	snortAdministrativo.conf	
20:15 – 00:00	snortTodos.conf	

Tabla 3 - Distribución horaria de diferentes configuraciones

3.5 Entorno gráfico

El software Snort está pensado para trabajar a nivel de servidor y en comunicación con otros elementos del entorno, pero no para la gestión diaria de los eventos por parte de un administrador humano. Debido a esta política no existe un gestor oficial de eventos y alarmas gráfico que permita un trabajo ágil y cómodo a los responsables de gestionarlo, lo cual ha hecho

que, gracias a una política de licencias abierta, se hayan desarrollado diferentes sistemas y entornos gráficos por parte de equipos independientes y sin conexión directa con Snort.

Aunque existen múltiples propuestas independientes de entornos gráficos de gestión de eventos (BASE (Johnson, 2016), Squert (Halliday, 2016), Snez (Guinter, 2016)…) hemos elegido Snorby (Ehrhorn, 2016) para nuestra prueba de concepto. Snorby está basado en una configuración de Snort en un sistema operativo Linux que vuelca las alarmas y eventos en una base de datos MySQL para su posterior consulta y gestión desde un entorno web limpio y sencillo.

Si bien existen instrucciones de instalación (Modesto Rossi, 2012) paso a paso para poder trabajar con Snorby para esta prueba de concepto hemos optado por utilizar una distribución Ubuntu ya configurada: Snorby Security Distribution (Bailey, 2013). Esta distribución como tal dejó de actualizarse en 2013 y utiliza Ubuntu 8.0.4 pero permite poder trabajar con Snorby en pocos minutos. Una vez instalado tan sólo debemos realizar las actualizaciones de los componentes, tanto de sistema operativo como de software, para obtener un sistema más moderno.

El entorno web de Snorby está basado en un cuadro de mandos (*Dashboard*) donde podemos obtener de un vistazo rápido el estado del sistema y las alarmas detectadas categorizadas por su criticidad. En esta misma página podemos ver una serie de estadísticas (número de eventos, número tipos de eventos, etc.) que permiten hacernos una idea de la situación.

En la pestaña *Events* podremos obtener toda la información sobre los eventos detectados por el sistema (evento, criticidad, fecha y hora del suceso, direcciones IP origen-destino del tráfico, etc.). En este mismo apartado podremos desactivar aquellos que ya hagamos gestionado o marcar como importantes aquellos que requieran una dedicación especial.

El sistema trae dos informes predefinidos para informarnos sobre todos los eventos, en la pestaña *Reports*, que pueden ser útiles tanto para los técnicos como para mostrar a los responsables la efectividad del sistema.

Se trata pues de un entorno sencillo y rápido de consultar que permite a los técnicos estar informados de las alertas detectadas de forma cómoda.

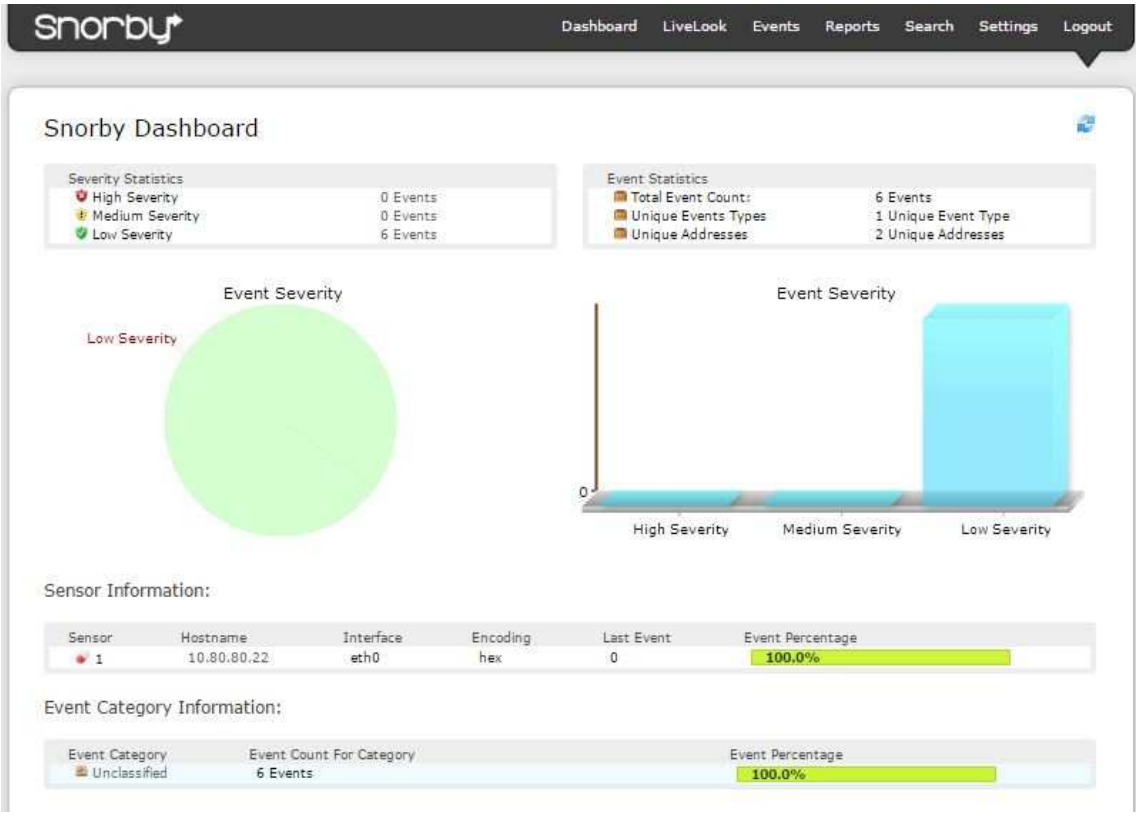

Ilustración 16 - Pantalla inicial de Snorby

4. Conclusiones

En esta memoria hemos intentado plasmar una solución válida y operativa para una necesidad específica que afronta la empresa que ha solicitado la consultoría. No hemos realizado un montaje estándar de un sistema IDS Snort, sino que hemos buscado la adaptación del sistema a las necesidades reales expresadas por la empresa.

Ante la imposibilidad de diferenciar técnicamente qué accesos a la información empresarial son legítimos y cuáles no hemos optado por una diferenciación horaria en el comportamiento del sistema Snort de tal forma que activamos una serie de firmas de detección cuando ha terminado el horario laboral. Aún con esta diferenciación de comportamiento por franja horaria hemos previsto una serie de conexiones que, por su poca habitualidad, pueden considerarse por sí mismas indicios de un intento de extracción de información por lo que son controladas en todo momento por el sistema.

Debido a los antecedentes hemos propuesto una configuración que envía un duplicado de los eventos a un servidor remoto. Gracias a esta configuración específica se mantendrá una copia de los datos en un servicio no gestionado por los propios técnicos con lo que se refuerza la posibilidad de detección de cualquier evento que lleva a pensar en una extracción no autorizada de información.

Finalmente hemos propuesto un sistema gráfico de gestión de eventos para reducir al mínimo posible el impacto sobre la carga de trabajo de los técnicos informáticos que gestionan y mantienen la infraestructura tecnológica de la empresa.

Hemos realizado esta prueba de concepto instalando el software Snort en local en servidores que simulan los dos entornos existentes en la empresa, pero si tras este proyecto se pretende

realizar una implantación en el entorno productivo real de la empresa habrá que reevaluar las características de la infraestructura física para decidir la mejor forma de instalación. Aunque lo más probable es que una solución estándar, con un equipo independiente escuchando el tráfico de red, sea una buena opción habría que analizarlo con los datos reales.

Dada la posibilidad de creación y adaptación de las firmas de detección es de esperar que con una instalación en entornos productivos se creen nuevas reglas para detectar situaciones no contempladas en esta prueba de concepto.

Otra mejora que ha quedado fuera de este proyecto es cómo proteger los ficheros de configuración y los ficheros de firmas del sistema Snort para detectar si han sido modificados antes de su última ejecución. Así podríamos detectar cualquier intento de cambio de comportamiento del sistema que podría ser un indicio de un intento de extracción de información o de cualquier otro ataque a la empresa.

Bibliografía

Bailey, P. (8 de julio de 2013). *Snorby Security Distribution - Homepage*. Obtenido de https://sourceforge.net/projects/spsa/

Barjatiya, S. (2 de junio de 2016). *Suarabh Barjatiya - Snort general rule options*. Obtenido de https://www.sbarjatiya.com/notes_wiki/index.php/Snort_general_rule_options#sid

Becker, A. (2 de junio de 2016). *HeidiSQL Homepage*. Obtenido de http://www.heidisql.com/

Caswell, B., & Houghton, N. (2 de junio de 2016). *Snort.org - Rule Docs*. Obtenido de https://www.snort.org/rule_docs/1-1775

Ehrhorn, G. (1 de junio de 2016). *Snorby - Homepage*. Obtenido de https://github.com/Snorby/snorby

Guinter, G. (1 de junio de 2016). *Snez - Homepage*. Obtenido de http://home.ptd.net/~gguinter/geneguintercom/SNEZ/SNEZ.html

Halliday, P. (1 de junio de 2016). *Squert - Homepage*. Obtenido de http://www.squertproject.org/

Johnson, K. (2 de junio de 2016). *BASE - Homepage*. Obtenido de http://base.professionallyevil.com/

Modesto Rossi, A. (27 de febrero de 2012). *Snorby - Instrucciones de instalación*. Obtenido de https://github.com/Snorby/snorby/wiki/Ubuntu-11.10-by-Andrea-Modesto-Rossi

Snort. (1 de junio de 2016). *FAQ - Snort.org*. Obtenido de https://www.snort.org/faq/readme-variables

Snort. (2 de junio de 2016). *Manual de Snort - Creación de regas*. Obtenido de http://manual-snort-org.s3-website-us-east-1.amazonaws.com/node27.html

Snort. (1 de junio de 2016). *Manual Snort - Definición y configuración de umbrales en firmas de detección*. Obtenido de http://manual-snort-org.s3-website-us-east-1.amazonaws.com/node35.html

Snort.org. (18 de junio de 2016). *Snort Users manual.* Obtenido de https://www.snort.org/documents/snort-users-manual

SolarWinds. (1 de junio de 2016). *Página oficial servidor Kiwi Syslog Server*. Obtenido de http://www.kiwisyslog.com/

Toro, L. (2 de junio de 2016). *DesdeLinux - Cron & crontab, explicados*. Obtenido de http://blog.desdelinux.net/cron-crontab-explicados/

VirtualBox. (4 de junio de 2016). *VirtualBox - Homepage*. Obtenido de https://www.virtualbox.org/

Wireshark. (2 de junio de 2016). *Página oficial de Wireshark*. Obtenido de https://www.wireshark.org/

Glosario

En este glosario detallamos algunas definiciones y conceptos utilizados durante la elaboración de este documento.

- **crontab:** En los sistemas tipo Unix un cron es un administrador en segundo plano que ejecuta procesos a intervalos regulares. La especificación de qué procesos y a qué hora deben ejecutarse se especifica en el fichero crontab.
- **DDoS:** Se trata de un ataque tipo **DoS** realizado desde diferentes orígenes. Viene de la expresión *Distributed Denial of Service*.
- **DoS:** Estas siglas inglesas (*Denial of Service*) hacen referencia a los ataques que buscan provocar que los usuarios legítimos no accedan a un servicio mediante la saturación de los servidores de forma que se impida dar correcto servicio a los clientes legítimos.
- **Escritorio Remoto:** Ver **RDP**.
- ***Firewall*:** Aunque en castellano se debería utilizar la palabra cortafuego la verdad es que el anglicismo se ha impuesto. Un cortafuego es un sistema diseñado para bloquear los accesos no autorizados mientras se permiten las comunicaciones autorizadas.
- **FTP:** Protocolo para la transmisión de archivos a través de una red IP. Viene del nombre inglés *File Transfer Protocol*.
- **IDS:** Los sistemas de detección de intrusos (*Intrusion Detection System*) se basan en el análisis de tráfico y la detección de patrones conocidos para activar alarmas antes un comportamiento anómalo. A diferencia de los IPS no hay actuación automática sobre el tráfico, sino que tan sólo informan de las situaciones detectadas.
- **IPS:** Los sistemas de prevención de intrusos (*Intrusion Prevention System*) monitorizan y bloquean activamente, a diferencia de los IDS, aquel tráfico que no cumple las reglas de acceso o no siguen el protocolo estándar.
- **Kiwi Syslog Server:** Se trata de un servidor de Syslog fácil de usar y configurar.
- **MySQL:** MySQL es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual GPL/Licencia comercial por Oracle Corporation y está considerada como la base datos *open source* más popular del mundo.
- **RDP:** El protocolo *Remote Desktop Protocol*, propietario de Microsoft, permite la comunicación en la ejecución de una aplicación entre un terminal cliente y un servidor. Se basa en el envío de la información gráfica que debería presentarse en pantalla a través de la red.
- **SMB2:** El protocolo *Server Message Block* (*SMB*) permite compartir archivos, impresoras, etcétera, entre nodos de una red de computadoras que usan el sistema operativo Microsoft Windows.
- **SSH:** El protocolo Secure SHell permite la conexión a un intérprete de comandos de una máquina remota.
- **Syslog:** Se conoce por este nombre tanto al protocolo de comunicaciones que permite el envío de mensajes de registro sobre una red IP como a las aplicaciones, ya sean clientes o servidores, que lo utilizan.
- **VirtualBox:** Es un software de virtualización que permite la creación de máquinas virtuales operativas. Actualmente el desarrollo está liderado por Oracle.
- **Wireshark:** Wireshark, antes conocido como Ethereal, es un analizador de protocolos utilizado para realizar análisis y solucionar problemas en redes de comunicaciones.

Anexo – Ficheros de firmas

El contenido del fichero *revisarSiempre.rules* es el siguiente:

```
#########################
## REGLAS REVISAR 7x24 ##
#########################

## ACCESO SSH ##
alert tcp any any -> 10.80.80.87 22 (msg:"UOC - Conexion SSH";  threshold:type limit, track
by_src, count 1, seconds 120; sid:1000001; rev:1;)
## ACCESO RDP ##
alert tcp any any -> 10.80.80.80 3389 (msg:"UOC - Conexion RDP";  threshold:type limit, track
by_src, count 1, seconds 120; sid:1000002; rev:1;)
## TRAFICO HTTP SALIENTE ##
alert tcp 10.80.80.80 any -> any 80 (msg:"UOC - HTTP saliente";  threshold:type limit, track
by_dst, count 1, seconds 120; sid:1000003; rev:1;)
## TRAFICO HTTPS SALIENTE ##
alert tcp 10.80.80.80 any -> any 443 (msg:"UOC - HTTPS saliente";  threshold:type limit, track
by_dst, count 1, seconds 120; sid:1000004; rev:1;)
## TRAFICO FTP SALIENTE ##
alert tcp 10.80.80.80 any -> any 21 (msg:"UOC - FTP saliente";  threshold:type limit, track
by_dst, count 1, seconds 120; sid:1000005; rev:1;)
## ACCESO root MYSQL ##
alert tcp any any -> 10.80.80.87 3306 (msg:"MYSQL root login attempt"; content:"root|00|";
classtype:protocol-command-decode; sid:1775; rev:2;)
```

El contenido del fichero *revisarAdministracion.rules* sería el siguiente:

```
############################
## REGLAS ADMINISTRACION ##
############################

## ACCESO A DIRECTORIO RecursosHumanos ##
alert tcp any any ->  10.80.80.80 445 (msg:"UOC - Acceso directorio Recursos Humanos";
content:"R|00|e|00|c|00|u|00|r|00|s|00|o|00|s|00|H|00|u|00|m|00|a|00|n|00|o|00|s
"; threshold:type limit, track by_src, count 1, seconds 120; sid:1000012; rev:1;)
## ACCESO A DIRECTORIO Facturacion ##
alert tcp any any ->  10.80.80.80 445 (msg:"UOC - Acceso directorio Facturacion";
content:"F|00|a|00|c|00|t|00|u|00|r|00|a|00|c|00|i|00|o|00|n"; threshold:type limit,
track by_src, count 1, seconds 120; sid:1000013; rev:1;)
```

A continuación, indicamos el contenido del fichero *revisarProductivo.rules*:

```
########################
## REGLAS PRODUCTIVO ##
########################

## USO APLICACION WEB-MYSQL ##
alert tcp $FRONTALES_WEB any -> 10.80.80.87 3306 (msg:"UOC - Acceso MySQL";
sid:1000006; threshold:type limit, track by_src, count 1, seconds 120; rev:1;)
```