UNIVERSIDAD DE ANTIOQUIA

Desarrollo de una estrategia híbrida para la gestión de alarmas en sistemas detectores de intrusos basados en firmas

Autor(es)
Francisco Javier Muñoz Cortés

Universidad de Antioquia
Facultad de Ingeniería, Departamento de Electrónica y Telecomunicaciones
Medellín, Colombia
2019

Desarrollo de una estrategia híbrida para la gestión de alarmas en sistemas detectores de intrusos basados en firmas

Francisco Javier Muñoz Cortés

Informe de práctica o monografía o investigación o tesis o trabajo de grado
como requisito para optar al título de:
Magister en Ingeniería de Telecomunicaciones

Asesores (a) o Director(a) o Co- Directores(a).
Ph.D Natalia Gaviria Gómez

Francisco Javier Muñoz Cortés, Ingeniero de Telecomunicaciones

Universidad de Antioquia
Facultad de Ingeniería, Departamento de Electrónica y Telecomunicaciones
Medellín, Colombia
2019.

Tabla de contenido

CAPITULO 1

Descripción del Proyecto

1. Introducción

El incremento de los ataques por parte de asociaciones criminales a la infraestructura crítica cibernética ha hecho que se haya volcado un interés masivo en desarrollar herramientas y estrategias que contribuyan al objetivo de reducir y neutralizar el impacto de estos. Uno de los principales bloques de construcción de estas estrategias de defensa ante ciberataques son los sistemas detectores de intrusos (IDS). Dentro de los diferentes tipos de IDS están los IDS de red (NIDS) basados en firmas que son los más usados comercialmente [1]. Los sistemas de detección de intrusos basados en firmas usan una base de datos de firmas de ataques conocidos y generan una alarma cuando el tráfico de la red coincide con cualquier regla. Entiéndase firmas como un conjunto de reglas que un IDS usa para detectar actividades intrusivas.

Las reglas que se cargan en la base de datos del IDS son frecuentemente lo más general posible, tal que una actividad indicando una remota posibilidad de ataque dispare una alerta. Normalmente, cuando un IDS se activa con un conjunto de reglas predeterminado, se generan grandes cantidades de alertas y registros. Estos registros ocupan espacio en el disco, las consultas a la base de datos de alarmas se vuelven lentas y a parte el análisis se torna complejo debido a la gran cantidad de alarmas no relevantes.
Por lo tanto, se convierte en una responsabilidad para el equipo analista de seguridad monitorear las alarmas generadas por el IDS para identificar aquellas que conducen a un ataque real del enorme número de falsas alarmas [2]. El analista deberá utilizar su experiencia para encontrar los eventos que son de interés. En algunos casos, si se puede identificar un patrón de falsos positivos o una fuente de falsos positivos, se podrán efectuar cambios en las políticas del IDS.

Uno de los grandes problemas, y que resulta ser factor decisivo para la línea de ruta marcada con este trabajo, es que derivado de la gran cantidad de falsas alarmas se hace totalmente complejo utilizar los eventos generados en tiempo real por el IDS para impulsar sistemas de respuesta automatizados, como lo sería la reconfiguración de un firewall, enrutador o cualquier otro dispositivo de red. El uso de este tipo de enfoque exigiría de un filtrado extremadamente intenso para limitar interrupciones no deseadas en la red.
Como se mencionó recién, uno de los mecanismos usualmente más usados por los analistas de seguridad para limitar el número de falsos positivos consiste en sintonizar de manera inteligente las firmas de la base de datos del IDS entendiendo el ambiente que están protegiendo, es decir, la red subyacente. Una vez se limita el número de falsos positivos mediante esta sintonización, la organización debe determinar un proceso para tomar acciones en las alertas restantes basado en el riesgo que estas representan. Lo anterior involucra la determinación de indicadores de compromiso que puedan ser usados para identificar aquellas alertas que representan el mayor riesgo y abordarlas de manera oportuna [4].
Estos indicadores de compromiso permiten determinar la importancia de una alerta mediante la asignación de un puntaje. Las técnicas de priorización de alarmas califican las alarmas con base en una post evaluación según el interés de la alarma basado en la amenaza que representen a los activos críticos de la red. En la literatura se han propuesto diversas técnicas de priorización de alarmas donde se propone la generación de un puntaje para cada alarma de acuerdo a diferentes parámetros definidos

por usuario como criticidad de las aplicaciones, cantidad de interés en un tipo de ataque, severidad del ataque, importancia del activo objetivo etc. [5][6].

Adicional a las técnicas de priorización, las técnicas de minería de alarmas se han convertido en una herramienta principal para evaluar la calidad de las alertas y tratar con el problema de falsos positivos en sistemas de detección de intrusos [7]. Estas técnicas usan atributos como direcciones IP, número de puertos, información de protocolos y demás, para minar un conjunto de alarmas y clasificarlas como alarmas de interés o falsas alarmas. Dentro de las diferentes técnicas de minería de datos, las técnicas de *clustering* han mostrado reducciones de hasta el 90% de falsas alarmas.
Dado que el ambiente de red es cambiante, se requiere de una estrategia que no solo sea eficiente en la reducción de las falsas alarmas y en la priorización de las alarmas de interés, sino también que persiga el carácter dinámico de la red y se adapte a estos cambios. Esta capacidad se puede conseguir buscando una combinación entre las técnicas de priorización, las técnicas de minería de alarmas y cualquier otra técnica de gestión de alarmas. Justamente las técnicas híbridas resultan de la combinación de las características de las técnicas mencionadas y esto hace que se perfilen como la solución central al problema de gestión de alarmas de IDS.

Teniendo en cuenta la necesidad que existe de tener una estrategia que capture el carácter dinámico de los eventos y amenazas de seguridad en la red y permita la gestión de alarmas en tiempo real [8], en esta tesis se presenta una estrategia escalable a ambientes de grandes volúmenes de datos para la identificación de alarmas asociadas con amenazas reales que contribuye a la disminución de los falsos positivos sin deterioros excesivos en la precisión y exactitud, que además presenta la característica de habilitar la correlación de eventos provenientes de sensores heterogéneos a través de la incorporación de Prelude SIEM [9] como gestor y normalizador de eventos, y que además incorpora información del contexto de la red proveniente del análisis de vulnerabilidades para tomar mejores decisiones. La estrategia usa estándares abiertos para correlacionar las alarmas del NIDS y la información proveniente del análisis de vulnerabilidades (VA).

1.1. Objetivo General

Desarrollar una estrategia híbrida para la gestión de alarmas en sistemas de detección de intrusos basados en firmas.

1.2. Objetivos Específicos

Analizar comparativamente las técnicas de priorización de alertas y minería de alarmas con base en sus resultados en reducción de falsas alarmas, precisión, sensibilidad y operación en tiempo real.

Definir e implementar una estrategia híbrida de priorización de alertas usando técnicas de minería de alarmas.

Evaluar la capacidad de la estrategia para identificar las alertas asociadas con amenazas reales y remover las alertas que no conducen a un riesgo factible.

1.3. Esquema del proyecto

El trabajo presenta la siguiente línea de ruta: en el capítulo 2 se abordan los sistemas detectores de intrusos basados en firmas pasando cuenta de las tecnologías más usadas en la industria. Además, se abordan las estrategias más usadas en un ambiente empresarial para gestionar las alarmas provenientes del IDS. Se remarca la importancia del análisis de vulnerabilidades y la correlación con

las alarmas. Por último, se introducen los sistemas para la gestión de eventos e información de seguridad SIEM.
En el capítulo 3 se analizan comparativamente las estrategias de gestión de alarmas que buscan minimizar el número de falsos positivos teniendo en cuenta la característica de reducción de falsas alarmas, los valores de precisión y sensibilidad, así como la posibilidad de operación en tiempo real, clasificando dentro de ese contexto las técnicas de priorización, minería de alarmas y también las técnicas híbridas.
En el capítulo 4 se describe e implementa la estrategia propuesta pasando por cada una de sus fases. En el capítulo 5 se describe el escenario de prueba para evaluar la estrategia propuesta. En el capítulo 6 se presentan y analizan los resultados y en el capítulo 7 se remarcan las conclusiones del trabajo y se define el trabajo a futuro.

CAPÍTULO 2

Sistemas detectores de intrusos basados en firmas y gestión de alarmas

En este capítulo se introducen los sistemas detectores de intrusos basados en firmas y se describe la relación entre el análisis de vulnerabilidades y la gestión de las alarmas como mecanismo para la reducción de falsas alarmas. En el capítulo 3 se ahonda en las diferentes estrategias para la gestión de alarmas registradas en el estado del arte.

Los sistemas detectores de intrusos (IDS) son equipos perimetrales o internos a la red que contribuyen en su plan de defensa detectando esencialmente patrones de tráfico que estén asociados con acciones maliciosas por parte de atacantes. En breve se comentan los tipos de IDS más comunes.
Los *IDS de red (NIDS)* son dispositivos que monitorean y analizan el tráfico de la red en búsqueda de accesos no autorizados. Las implementaciones de NIDS típicamente incluyen más de un sensor en la red para tener una visión general de todo el tráfico que la atraviesa y se dispone de una o más consolas de gestión para el analista de seguridad. Los sensores pueden operar en dos modos [10]: *en línea y pasivos*.
En el modo de operación *en línea* el tráfico que está siendo monitoreado debe pasar a través del sensor. Este tipo de sensores se usa generalmente para bloquear el tráfico que se ha detectado como malicioso, operando en modo de detección y prevención.
Por otro lado, los sensores *pasivos* monitorean una copia del tráfico que pasa por la red, no el tráfico original. Se dice que la operación del sensor en modo pasivo es más eficiente que en modo de operación en línea ya que no añade un paso extra en la gestión del paquete que puede causar retardo.

Más allá de su modo de operación los NIDS también se clasifican según su método de detección.
Los *NIDS basados en firmas* usan una base de datos de ataques conocidos para buscar coincidencias en los patrones de tráfico atravesando la red.
En el caso de los *IDS basados en anomalías*, se dispone de un perfil común o línea base que permite al IDS aprender la variación del tráfico en un periodo de tiempo. Luego del periodo de aprendizaje, se estudia estadísticamente el tráfico recolectado en un periodo de tiempo y se crea un perfil base con base en este estudio. El tráfico que esté por fuera del perfil aprendido se clasificará como tráfico sospechoso.
A continuación, se introducen dos de los NIDS basados en firmas de estándar abierto más usados, a saber, Snort y Suricata. Adicionalmente se introducen otras soluciones de proveedores específicos como FireEye y Palo Alto Networks.

2.1. Snort

Snort [11] es un IDS/IPS de estándar abierto que fue lanzado originalmente en 1998 por Martin Roesh como una herramienta para inspeccionar el tráfico de la red. Luego de su lanzamiento, su creador se dio cuenta que Snort también podría ser configurado para buscar patrones en el tráfico y disparar alarmas en caso de coincidencia con estos patrones (reglas).
Snort se puede emplear como IDS, que es su característica principal, y como sistema de prevención de intrusos (IPS). En el primer caso Snort inspecciona el tráfico vía puertos espejos o *taps* de red dedicados, generando alertas para el tráfico que se identifique como anómalo o malicioso. Por otro

lado, en modo IPS, donde el tráfico debe pasar por las tarjetas de red del sensor, Snort puede cortar el tráfico que se identifique como malicioso.
Snort se puede configurar para manejar fragmentación de paquetes IP, reensamblaje de flujo de sesión TCP, etc. Estas características en Snort se conocen como características de normalización de tráfico de red y se operan de manera automática antes de evaluar el tráfico contra cualquier regla para garantizar que Snort vea el flujo de tráfico completamente reensamblado, decodificado y desofuscado. La mayoría de las reglas proporcionadas en el conjunto de reglas TALOS provistas en *snort.org* tienen mensajes de alerta bastante intuitivos que describen el tráfico que activaron y / o los metadatos de referencia que apuntan a sitios web a los que puede acceder para obtener más información.

Las reglas para los sensores IDS se pueden gestionar modificando el archivo de configuración manualmente o usando comandos especiales. Snort soporta varios formatos de salida para las alertas; se pueden enviar mediante mensajes *syslog,* guardar el tráfico que dispara las alertas en formato *pcap* [12] y por último en formato *unified2* (registro binario). Se recomienda usar el formato *unified2* ya que es más fácil y más rápido para Snort emitir alertas en este formato.
Las alarmas en este formato se pueden parsear a diferentes formatos, SQL por ejemplo, para almacenamiento en diferentes tipos de motores de bases de datos (MSSQL, Oracle, Mysql, etc.). Hay otro parser conocido como *u2json21* el cual convierte el formato binario unified2 a JSON para integración como SIEMs como Splunk [13] y stacks de fuente abierta ELK [14] (Elasticsearch, Logstash, Kibana).
Uno de los inconvenientes con Snort tiene que ver con la escalabilidad. Por defecto, Snort no es multihilo. Para escalarlo de una manera efectiva una vez se superen los 1 Gbps de rendimiento en la inspección del tráfico de la red, se deben correr múltiples procesos de Snort.

2.2. Suricata

Suricata [15] es un IDS/IPS de fuente abierta que fue lanzado inicialmente en 2009, como un producto de la Fundación Abierta de Seguridad de la Información (OISF por sus siglas en inglés). Muchas de las características de Suricata son casi idénticas a las de Snort. En primer lugar, Suricata tiene un lenguaje de reglas similar, capacidades de normalización similares (desfragmentación IP, reensamblaje de flujos TCP, el manejo de segmentos duplicados/superpuestos y demás) e incluso soporta formatos de salida de Snort por temas de compatibilidad con la mayoría de las aplicaciones web y herramientas diseñadas para ser usadas con Snort (por ejemplo, el formato de salida unified2).

Sin embargo, pese a las similitudes, hay ciertas características de Suricata que superan las de Snort. Por defecto, Suricata es multihilo, lo que significa que es mucho más sencillo escalar Suricata con el objetivo de inspeccionar tráfico en redes Gigabit. Suricata también admite el uso de tarjetas gráficas para ayudar a descargar y escalar la inspección de la red. Además de la escalabilidad, Suricata tiene la capacidad de registrar muchos más datos que Snort. Por defecto, Suricata admite el registro de consultas DNS, encabezados HTTP y solicitudes y respuestas de cuerpo, datos similares a Netflow, información de certificados SSL y también puede extraer archivos de flujos de red y almacenarlos en disco (a diferencia de Snort) [10].

2.3. Soluciones propietarias

Además de Snort y Suricata, existen otras soluciones IDS/IPS propietarias, como FireEye, Palo Alto Networks, Checkpoint IPS y muchas otras. La solución IDS de FireEye es esencialmente una implementación de Snort con una configuración que no es modificable por el usuario, mientras que PAN, Checkpoint y otros proveedores usan sus propias soluciones. La ventaja de la mayoría de estas soluciones es que, por lo general, son de nivel profesional, están respaldadas por compañías de

seguridad relativamente grandes y cuentan con el respaldo adecuado para que los clientes puedan obtener ayuda con problemas de implementación, falsos positivos y/o errores, según sea necesario. Dado que los productos están sujetos a control de calidad y, en la mayoría de los casos, el hardware se prueba con calidad antes de ser vendido, se sabrá exactamente (con algunas leves variaciones) cuánto tráfico pueden manejar los sensores.
Por otro lado, el detalle con Snort y/o Suricata es que se conoce cómo funciona la detección y cómo se ven las firmas/reglas. Con estas plataformas de IDS alternativas, no se tiene certeza acerca de cómo se está realizando la detección o cómo se ven las firmas. Esto son sistemas de código cerrado. Con los productos de código abierto, el software en sí es gratuito, pero también lo es el soporte.

¿Qué plataforma es la mejor?

En [10] se menciona que cada solución tiene un costo asociado, ya sea que se trate de un gasto operativo que requiera tiempo y mano de obra para mantener, o un gasto de capital que requiera fondos importantes para comprar una solución que tenga soporte integrado. La recomendación que se hace en [10] es que, si se tuviera que desarrollar una nueva implementación de un gestor de seguridad de red (NSM por sus siglas en inglés) desde cero, y sólo se tuvieran implementaciones de IDS de código abierto de las que se pudiera elegir, se usaría Suricata debido a la gran cantidad de funciones y a temas de escalabilidad. Por otra parte, si se dispone de un presupuesto para soluciones comerciales, se propone analizar los informes de Gartner y/o NSS Labs para ver cómo se apilan los diferentes proveedores de NSM en términos de rendimiento y/o precio.
A continuación, se hace una comparación cualitativa de los IDS de estándar abierto (tabla 1) basado en diferentes factores de interés [16]. Asimismo, en la tabla 2 se comparan tres de los IDS propietarios más usados en la industria con base en las opiniones recogidas por diferentes tipos de usuarios y publicada por Gartner [17].

Tabla 1. Comparación basada en diversos factores de IDS de estándar abierto de interés.

Parámetros	**Snort**	**Suricata**	**BRO** [18]
Plataforma soportada	Win, MacOS, Unix	Win, MacOS, Unix	Unix, MacOS
Licencia	GNU GPL V2	GNU GPL V2	BSD
Característica IPS	Sí	Sí	No
Firmado PGP (Pretty Good Privacy)	Sí	No aplica	No
Soporte en redes de alta velocidad	Medio	Alto	Alto
GUI de configuración	Sí	Sí	No
Análisis off-line	Sí, para múltiples archivos	Sí, para un solo archivo	Sí, para un solo archivo
Hilos	Hilo único	Multihilo	Hilo único
IPv6	Sí	Sí	No
Instalación e implementación	Fácil	Fácil	Difícil

Tabla 2. Comparación basada en diversos factores de IDS de proveedores propietarios [17].

Parámetros	**Darktrace** (Darktrace)	**FireEye** (FireEye Network Security [NX])	**Cisco** (Cisco Firepower, Virtual Next-Generation IPS [NGIPSv] para VMware)
Calificación general de pares / Disponibilidad para recomendar	**4.7** (39 op.) / **88% Sí** (40 op.)	**4.4** (37 op.) / **70% Sí** (37 op.)	**4** (54 op.) / **67 % Sí** (54 op.)
Capacidades del producto	**4.7** (39 op.)	**4.4** (37 op.)	**4.2** (54 op.)
Experiencia de usuario			
Evaluación y contratación / Flexibilidad de precio	**4.5** (37 op.) / **4.2** (36 op.)	**4.2** (35 op.) / **3.8** (34 op.)	**4.2** (47 op.) / **3.8** (48 op.)
Integración e implementación / Facilidad en la implementación	**4.7** (39 op.) / **4.6** (38 op.)	**4.4** (37 op.) / **4.4** (37 op.)	**4** (53 op.) / **3.9** (53 op.)
Soporte y servicio / Puntualidad en la respuesta del proveedor / Calidad en el soporte técnico	**4.6** (20 op.) / **4.6** (35 op.) / **4.7** (35 op.)	**4.5** (11 op.) / **4.5** (37 op.) / **4.5** (37 op.)	**3.9** (24 op.) / **4** (48 op.) / **3.9** (52 op.)

La sigla op. se refiere a opiniones. De las dos tablas anteriores se puede notar que, según los parámetros de comparación usados, si se trata de un IDS de estándar abierto Suricata sería una buena elección considerando temas de escalabilidad. Por otro lado, si se trata de una solución propietaria y basado en las opiniones de los usuarios registrada en la tabla 2, Darktrace sería una buena elección.

Más allá de la herramienta que se use, que, por supuesto tendrá un peso importante a la hora de elegir la estrategia de gestión de alarmas, es cierto que el tema central del proyecto es la gestión de las alarmas. Por lo tanto, es importante considerar el formato en que puedan presentar las mismas. En la siguiente sección se analiza el formato IDMEF [19] que es el formato estándar elegido por la IETF [20] para la representación de eventos de detección de intrusos.

2.4. Formato de las alarmas y/o eventos de seguridad relacionados con las alarmas

Debido a que en una red de telecomunicaciones pueden coexistir diferentes tipos de NIDS, es importante tener una representación estándar y unificada de la información de seguridad, no sólo de las alarmas generadas por los IDSs, sino también de los eventos de seguridad como logs del sistema, logs de firewalls etc. La razón es evidente: hacer más sencilla la tarea de gestionar la información.
En [21] se hace una evaluación de los formatos y protocolos para el intercambio de eventos de seguridad en el contexto de redes de alta velocidad. Los formatos para el intercambio de eventos de seguridad se pueden categorizar en cuatro grupos: expresiones-S, basados en XML, basados en mensajes MIME-message y syslogs. En la tabla 3 se muestra el resumen de la evaluación llevada a cabo en [21].

Tabla 3. Evaluación de los diferentes formatos para el intercambio de información de eventos de seguridad. Tomada de [21].

Alto: H, Medio: A, Bajo: L										
Criterio	**CIDF [22]**	**IODEF [23]**	**CAIF [24]**	**IDMEF**	**ARF [25]**	**CEE [26]**	**x-arf [27]**		**Syslog**	
							V0.1	**V0.2**	**RFC 3164 [28]**	**RFC 5425 [29]**
Interoperabilidad	L	L	L	L	H	H	H	H	H	H
Extensibilidad	H	H	H	H	H	H	H	H	H	H
Escalabilidad	L	L	L	L	L	L	L	L	L	L
Agregabilidad	L	L	H	A	L	L	L	H	L	L
Independencia de protocolo	L	A	H	A	H	A	H	H	H	H
Lectura de máquina	H	H	H	H	H	H	H	H	L	H
Integridad y autenticidad	L	L	L	L	L	L	L	H	L	L
Confidencialidad	L	L	L	L	L	L	L	H	L	L
Aplicación práctica	L	A	A	A	A	L	A	A	H	H

Para un mayor detalle acerca de los criterios de evaluación para los diferentes formatos se sugiere al lector chequear el artículo del cuál se desprenden los resultados, así como las referencias a cada uno de los formatos. Los formatos basados en MIME son ARF y x-arf. Entre los basados en expresiones-S está el formato CIDF.

Los autores encontraron que para el uso de datos basados en flujo se requiere que los formatos basados en XML como lo son IODEF, CAIF y IDMEF usen un nuevo esquema o extiendan el campo *AdditionalData*. Adicionalmente, a partir de la tabla se evidencia que la mayoría de formatos son leíbles por máquinas, lo que asegura que los eventos de seguridad se pueden manejar automáticamente.

Con relación a la interoperabilidad con los datos basados en flujos, se menciona que los formatos de intercambio más adecuados para este uso son ARF, CEE, x-arf y Syslog. Para intercambiar información sensible se recomienda usar el formato x-arf que provee mecanismos para firmar o cifrar los eventos de seguridad.

En el presente trabajo se usa el formato IDEMF debido a su uso e implementación en herramientas de estándar abierto (Prelude, Snort, Suricata, OSSEC etc.). El modelo de datos IDMEF se implementa utilizando una Definición de tipo de documento (DTD) para describir documentos en lenguaje de marcado extensible (XML). IDMEF también es una representación orientada a objetos y un modelo de Lenguaje de modelado unificado (UML), como se muestra en la figura 1. Por otro lado, en la tabla 4 se muestra la descripción de cada clase en IDMEF [30].

Ahora, la cuestión que surge es ¿cómo se gestionan las alarmas provenientes de un IDS y en todo caso cómo se aborda el problema de las falsas alarmas? Las siguientes secciones introducen los métodos más comunes empleados para la gestión de las alarmas de un IDS.

2.5. Diagramas de red y/o gestión de activos

Una de las formas intuitivas para reducir el número de alarmas regulares que dispara el IDS es ajustando las reglas del IDS. Para hacer una interpretación adecuada de las reglas del IDS es

importante tener una buena gestión de los activos de red. Se hace necesario tener información acerca de los sistemas operativos y de los servicios corriendo en la red. Esta es justamente la salida de los sistemas usados para la gestión de activos y detección de servicios.

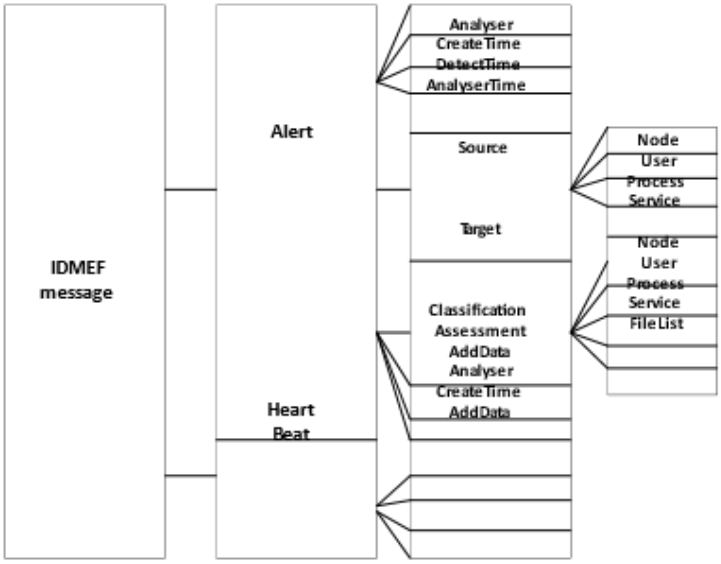

Figura 1. Estructura del formato de mensaje para eventos de detección de intrusos IDMEF. Tomado de [30].

Tabla 4. Descripción de las clases del formato IDMEF [30].

Clase	Descripción
Alert	Dependiendo del sensor, un mensaje de alerta puede corresponder a múltiples o a un solo evento detectado. Las alertas ocurren asíncronamente en respuesta a eventos.
Analyser	Información de identificación para el sensor que originó la alerta.
CreateTime	El tiempo en que la alerta fue creada.
DetectTime	El tiempo en que se detectaron los eventos que dispararon la alerta.
AnalyserTime	Tiempo actual en el analizador.
Source	Fuente de eventos que dispararon las alertas
Target	Objetivo de los eventos que dispararon las alertas.
Clasification	El nombre de la alerta
Assessment	Información acerca del impacto del evento, acciones tomadas por el analizador en respuesta al evento y la certeza del analizador en su evaluación.
AddData	Información incluida por el analizador que no se acomoda en ninguna de las otras clases del modelo de datos.
Heartbeat	Indica el estado actual del analizador. Se envía en periodos regulares de tiempo.
Analyser	Información de identificación para el analizador que originó el heartbeat.
CreateTime	El tiempo en el que el heartbeat fue creado.
AddData	Descripción similar al anterior AddData.

Entre las herramientas más usadas con este propósito están P0F [31], prads [32], BRO, KVM o IPMI. Independientemente del tipo de herramienta que se use para la gestión de los activos y la detección de servicios se recomienda que la información recolectada sea actualizada regularmente [10].
De igual manera, la información proveniente de soluciones de seguridad basadas en host como logs del sistema (syslog, dmesg, auth, secure), antivirus como Symantec [33], Eset [34], Sophos [35], Growdstrike [36], Carbon Black [37], Cylance [38], alertas de análisis de vulnerabilidades y demás, ayudan a mejorar la capacidad de respuesta ante las alertas del IDS, proporcionando información de contexto que ayuda a determinar la veracidad de las mismas.
La información de contexto puede ser obtenida a partir de los análisis de vulnerabilidades que se ejecutan en la red. En la siguiente sección se describe cómo la salida de los análisis de vulnerabilidades puede contribuir a determinar la veracidad de las alarmas disparadas por el IDS.

2.6. Análisis de vulnerabilidades y sistemas de detección de intrusos

Las herramientas para la evaluación de vulnerabilidades (conocidos como escáneres) realizan evaluaciones a los sistemas para determinar las vulnerabilidades explotables.
Los escáneres usan dos enfoques para llevar a cabo estas evaluaciones. El primer enfoque se sirve de los *mecanismos pasivos basados en hos*t los cuales inspeccionan los archivos de configuración del sistema en busca de configuraciones no adecuadas, los archivos de contraseñas y otros objetos del sistema en busca de violaciones de la política de seguridad [39]. El segundo enfoque se sirve de *mecanismos activos basados en la red* que recrean las secuencias de comandos de intrusión comunes, registrando su respuesta. El proceso anterior da una visión acerca del estado vulnerable de la red.
Aunque estos sistemas no pueden detectar de manera confiable un ataque en progreso, pueden determinar que un ataque es posible y, además, en ocasiones pueden determinar que ha ocurrido un ataque.

Las herramientas para el análisis de vulnerabilidades ejecutan un chequeo exhaustivo al sistema en busca de las exposiciones a vulnerabilidades de seguridad del sistema. Estas herramientas reportan entre otras cosas el número (referencia), la naturaleza y la severidad de estas exposiciones. En términos prácticos, los productos para el análisis de vulnerabilidades complementan los IDS ya que permiten al administrador del sistema operar en un modo proactivo encontrando y cerrando los agujeros de seguridad antes de que los atacantes los puedan usar.

2.6.1. Evaluación de vulnerabilidades

En [39] se proponen 4 enfoques para la evaluación de vulnerabilidades que registramos a continuación.

- **Evaluación basada en aplicación/servicio**

Este tipo de evaluación usa técnicas pasivas y no invasivas para chequear los ajustes y configuraciones en las aplicaciones que se descubren vulnerables. Entre entras técnicas están los escaneos pasivos que inspeccionan permisos de archivos, propietarios de archivos críticos, ruta de configuraciones etc.

- **Evaluación basada en el host**

Este tipo de evaluación de vulnerabilidades usa técnicas para chequear el sistema interno e incluye análisis de permisos de archivos y si se han aplicado parches de errores al sistema operativo. También se corren crackers de contraseñas contra los archivos de contraseñas para registrar contraseñas débiles. Este tipo de evaluaciones son muy precisas y en general se detectan vulnerabilidades que no

se exponen durante una evaluación basada en red. Adicionalmente, este método es específico de la plataforma donde se realice la evaluación por lo que se requiere una configuración precisa para cada tipo de host.

- **Evaluación basada en el objetivo**

Este tipo de técnicas verifica la integridad de los archivos y los datos del sistema, así como los objetos del sistema y sus atributos (por ejemplo, flujos de datos ocultos, bases de datos y claves de registro). Las herramientas de evaluación de vulnerabilidades basadas en objetivos utilizan sumas de comprobación criptográficas para hacer evidente la manipulación de objetos y archivos de sistemas críticos. Los algoritmos de resumen de mensajes se basan en funciones hash, que poseen la propiedad de que los cambios extremadamente sutiles en la entrada de la función producen grandes diferencias en el resultado. Esto significa que un cambio en un flujo de datos alimentando a un algoritmo de resumen de mensajes produce un gran cambio en la suma de comprobación generada por el algoritmo.

Cuando al aplicar la función hash se detectan cambios, el producto envía un mensaje al sistema de detección de intrusos que registra el problema con una marca de tiempo correspondiente al tiempo probable de alteración. Este proceso puede proporcionar un desencadenante de un registro para una alerta de intruso o puede servir como un hito para un investigador que realiza un seguimiento de los eventos que llevan a la alteración [39].

- **Evaluación basada en la red**

La evaluación de vulnerabilidad basada en la red utiliza técnicas activas e invasivas para determinar si un sistema determinado es vulnerable a un conjunto de ataques. En la evaluación basada en la red, se recrea una variedad de escenarios de ataque contra los sistemas objetivo y se analizan los resultados para determinar la vulnerabilidad del sistema al ataque. En algunos casos, la evaluación de red se usa para buscar problemas específicos de la red.

Asumiendo que aún no se ha sintonizado el conjunto de reglas del IDS a la red subyacente, este producirá una alarma ante cualquier ataque potencial. Es importante recibir todas las alertas, pero también lo es el hecho de hacer un procesamiento inteligente de las mismas. Por ende, al integrar la salida de la herramienta de análisis de vulnerabilidades se puede determinar si en efecto el sistema es vulnerable al ataque por el cual se disparó la alarma. Algunas de las ventajas de este enfoque tienen que ver con el hecho de que no depende de la plataforma donde se realice, por lo que es fácil de implementar. Sin embargo, al compararlo con la evaluación basada en host, se puede decir que es menos preciso dado que no considera vulnerabilidades de plataformas específicas [39]. Adicionalmente se puede ver afectado el rendimiento y la operación de la red ya que esta es una técnica de evaluación activa.

En general, se puede decir que los métodos de evaluación son complementarios ya que usan técnicas pasivas y activas y por ende habrá mayor precisión en los resultados del análisis. Habrá que considerar qué tan oportuno es usar las evaluaciones basadas en la red ya que como se mencionó anteriormente, en este enfoque se simulan los ataques para descubrir las vulnerabilidades y en términos de tráfico se puede ver reducido el rendimiento de la red.

2.6.2. Correlación de información de vulnerabilidades con alertas del IDS

Como se ha venido mencionando en los apartados anteriores, la información extraída a partir del análisis de vulnerabilidades resulta en una pieza clave para determinar la validez de una alarma que

se está disparando. La correlación de las alarmas con esta información habilita la toma de mejores decisiones de una manera automatizada.
No obstante, existe una limitación con respecto a este enfoque y tiene que ver con los conceptos de falsos positivos y falsos negativos tanto por parte del IDS como por parte del análisis de vulnerabilidades (AV). Igual que en el caso de los IDS, en las tecnologías para el AV puede haber una detección falsa de una vulnerabilidad, así como la no detección de la misma. En [40] se hace una categorización de los casos en los que se correlacionan las alarmas del IDS y la información del AV, que registramos a continuación.

- **Falsas alarmas provenientes del IDS con falsos positivos de vulnerabilidades**

En este caso, el IDS detecta un ataque falso contra el sistema y de manera análoga, del AV se detectan vulnerabilidades asociadas con el ataque, no obstante que el sistema no es vulnerable a el mismo. Dado que hay una coincidencia de la alarma con información de vulnerabilidades del sistema, esta se etiquetará como una alarma de alta calidad.
La manera de prevenir este escenario es sintonizando tanto el IDS como la solución para el AV. Este escenario se puede identificar buscando la relación entre el sistema operativo objetivo, la aplicación o servicio, el evento del IDS y la vulnerabilidad [40].

- **Falsas alarmas del IDS con vulnerabilidades falsas**

En este escenario se ignora la alarma dado que no hay información registrada de las vulnerabilidades asociadas. Se considera que este caso no es un problema.

- **Falsas alarmas del IDS con vulnerabilidades verdaderas**

En este escenario la mayoría de las alarmas serán desechadas ya que no hay vulnerabilidades que coincidan.

- **Falsos negativos del IDS con falsos positivos, falsos negativos y vulnerabilidades verdaderas**

En estos escenarios es difícil aplicar una correlación. Si el IDS puede ser eludido, no habrá eventos que correlacionar. En efecto, si el IDS puede ser eludido, todo el sistema será eludido.

- **Alertas verdaderas del IDS con vulnerabilidades falsas**

En este caso el IDS detecta un ataque real pero la herramienta usada para el AV ha detectado incorrectamente una vulnerabilidad relacionada con este ataque. Este escenario puede ser identificado cuando el evento del IDS se investiga y los sistemas objetivos se encuentran no vulnerables a este ataque.

- **Alarmas verdaderas del IDS con falsos negativos del AV**

En este caso se desechan alarmas que conducen a ataques reales dado que se está pasando por alto información relevante acerca de las vulnerabilidades. Este tipo de escenarios ocurren con cierta probabilidad en situaciones de días cero (*zero-day)* en el cual un servidor se ve comprometido con una vulnerabilidad que aún no ha sido publicada.

❖ **Alarmas verdaderas con vulnerabilidades verdaderas**

Este es el caso ideal para la correlación dado que el IDS detecta el ataque y el AV identifica correctamente que el sistema objetivo es vulnerable a esa forma de ataque específica.
En la tabla 5 se resume los casos posibles en la correlación de alarmas y vulnerabilidades, resaltando cuáles son los casos deseables y cuáles no.

Tabla 5. Resumen de los casos posibles en la correlación de alarmas e información de vulnerabilidades.

AV \ IDS	**Falsa alarma**	**Falso negativo**	**Alarma verdadera**
Vulnerabilidad falsa	Correlación falsa (caso no deseado)	No hay correlación (IDS eludido)	Correlación falsa (caso no deseado)
Falso negativo	No hay correlación (no problema)	No hay correlación (IDS eludido)	No hay correlación (caso no deseado)
Vulnerabilidad verdadera	No hay correlación (caso deseado)	No hay correlación (IDS eludido)	Hay correlación (caso deseado)
Negativo verdadero	No hay correlación (caso deseado)	-	-

2.6.3. Correlación y soluciones propietarias

Las soluciones propietarias tienen sus métodos específicos para relacionar información de eventos del IDS e información del AV. Ahora, cuando no se dispone de una fuente única para el AV y los eventos del IDS, es factible buscar una correlación flexible de estándares abiertos. Existe una gran variedad de estándares abiertos que se pueden usar para correlacionar diferentes IDS con diferentes tecnologías para el AV. Estas incluyen el programa de Enumeración de Vulnerabilidad Común de Mitre (*http://cve.mitre.org/),* los avisos CERT del Equipo de Respuesta a Emergencias Informáticas (*http://www.cert.org/),* la base de datos de *Bugtraq* (http://www.securityfocus.com/bid) y en algunos casos, se pueden utilizar los identificadores de vulnerabilidad de *Nessus* [40].

Una vez se corran los escáneres de vulnerabilidades, se extrae toda la información relevante asociada con las referencias mencionadas recién. Esta información se asociaría directamente con las referencias asociadas a los eventos del IDS. Sin embargo, muchos proveedores de IDS no publican directamente las reglas en un formato usable. Por supuesto que esto dificulta el poder establecer una relación entre las vulnerabilidades y los eventos disparados por el IDS.

En [40] se mencionan cuatro modelos de correlación de IDS/AV que referenciamos a continuación:

2.6.3.1. Políticas del IDS basadas en las vulnerabilidades

Este modelo usa la información obtenida a partir del AV para crear las políticas del IDS. La ventaja de esta solución es que los sensores IDS pueden correr más rápido. La ejecución con menos firmas es una buena manera de aumentar la velocidad de cualquier IDS de host o red. Adicionalmente se espera que haya menos falsos positivos. La desventaja de esta solución es que bajo esta configuración se puede desechar información de ataques que pueden resultar de interés.

De cualquier manera, dependerá de los analistas de seguridad si prefieren usar el enfoque de descartar información que no se vincule directamente con las vulnerabilidades detectadas o por el contrario tener la mayor cantidad de información posible que se vincule con intentos de intrusión.

2.6.3.2. Correlación de IDS/AV persistente

En este enfoque se mantiene una base de datos de las vulnerabilidades de la red, que a su vez se correlaciona con las alertas del IDS. Cuando se encuentra una coincidencia, se etiqueta la alarma en consideración como de alta prioridad. El sistema debería tener un método para consultar la base de datos de vulnerabilidades lo suficientemente rápido como para mantenerse al día con los eventos generados por el IDS. Uno de los beneficios de este método de correlación es que se pueden recibir eventos de diferentes IDS. Es muy probable que diferentes tecnologías IDS detecten ataques de diferentes maneras por lo que al disponer de más información se aumentan las posibilidades de que se produzca el escenario que encuentra un evento IDS válido dirigido a una vulnerabilidad válida conocida. Es importante tener la base de datos de información de vulnerabilidades actualizada para poder hacer una correlación acertada.

2.6.3.3. Correlación de IDS/AV en tiempo cercano

Este modelo de correlación es similar al anterior, pero no mantiene una base de datos permanente de vulnerabilidades. En lugar de ello, cada que ocurre un evento del IDS, la red es consultada de manera activa en búsqueda de información de vulnerabilidades. Este tipo de correlación tiene una desventaja en entornos de IDS de alta velocidad, donde las alertas del IDS se producen a una velocidad tal que no hay tiempo para detenerse y verificar la alarma.

2.6.3.4. Correlación de IDS/AV en tiempo real

En este tipo de correlación la información de vulnerabilidades se deriva en tiempo real. Si el NIDS puede derivar una vulnerabilidad (con una firma) será capaz de realizar la correlación de alarmas y vulnerabilidades en tiempo real. Sin embargo, esta técnica puede no ser tan efectiva ya que muchas vulnerabilidades no pueden derivarse de forma pasiva y deben requerir una interacción activa de una herramienta para el AV.

A partir de lo anterior puede decirse que, aunque cada método tiene su ventaja, es notorio que el método de correlación persistente da un grado más de fiabilidad al tener una base de datos actualizada con información de vulnerabilidades que ayudarán a tomar mejores decisiones para las alarmas generadas. Adicionalmente, se podría combinar este método con las políticas basadas en vulnerabilidades para aumentar la precisión y la calidad de las alarmas generadas.

Hasta aquí no se han mencionado las herramientas para realizar el análisis de vulnerabilidades, por lo que es tiempo de introducir dos de las más usadas, Nessus y Nmap.
Nessus [41] es un sistema activo de detección de vulnerabilidades de estándar abierto administrado por Tenable. Los scripts de Nessus ofrecen referencias a las bases de datos de Bugtraq y CVE, entre otras. Estos scripts se pueden usar para definir cómo descubrir vulnerabilidades y versiones de productos. Adicionalmente, Nessus también puede ser usado para establecer relaciones entre vulnerabilidades y productos.

Entre otros, Nessus permite detectar los siguientes tipos de vulnerabilidades [42]:

- Vulnerabilidades explotadas por hackers que tienen objetivos maliciosos
- Configuraciones erróneas (por ejemplo, parches faltantes)

- Contraseñas por defecto
- Denegación de servicios contra el stack TCP/IP

Nmap es un escáner de seguridad usado para descubrir hosts y servicios en una red de computadores y crear un mapa de la red. Algunas de sus características principales son [42]:

- Descubrimiento de hosts
- Escaneo de puertos
- Detección de versión de servicios
- Detección de sistema operativo

Entre otras, Nessus usa las bases de datos de Bugtraq y CVE para identificar las vulnerabilidades. Bugtraq [43] es una base de datos de vulnerabilidades conocidas. Contiene información detallada sobre la descripción de la vulnerabilidad, los productos a los que afectan, sus vulnerabilidades conocidas y las formas de prevenir estas vulnerabilidades

Diccionario CVE

CVE [44] es un diccionario que asigna a cada vulnerabilidad un número único y una descripción. El número de vulnerabilidad se separa en tres partes. La primera parte es el prefijo CAN o CVE para las vulnerabilidades candidatas y confirmadas, respectivamente. La segunda parte es el año en que se descubrió la vulnerabilidad. La última parte es un número secuencial que identifica las vulnerabilidades enumeradas durante un año. El diccionario CVE proporciona números de referencia a otras bases de datos de vulnerabilidades, pero no provee ninguna asociación entre vulnerabilidades y productos o entre explotaciones y vulnerabilidades [42].

Como se mencionó anteriormente sobre las versiones de IDS propietarios, cada uno de estos sistemas viene con una interfaz de usuario propia del proveedor. Lo anterior supone una dificultad a la hora de correlacionar alarmas provenientes de IDS de diferentes proveedores. Justamente para solventar esa dificultad es que aparecen los Sistemas de Gestión de Información de Seguridad (SIEM). Los SIEM recopilan eventos de diversas fuentes, cada uno de los cuales puede representar eventos utilizando un esquema específico del proveedor; normalizar estos esquemas dispares en una representación común; y almacenar estos eventos normalizados. Su motor de reglas dispara alertas de los eventos almacenados. Estas reglas a su vez permiten la correlación de eventos de diferentes sensores. Los sistemas SIEM también incluyen información contextual auxiliar, como información actualizada sobre los activos de la empresa que se pueden usar para escribir reglas customizadas y conscientes del contexto, con el fin de priorizar las alertas de interés.
La principal fortaleza de los sistemas SIEM es su capacidad de correlacionar registros de diversas fuentes utilizando atributos comunes para definir patrones y escenarios de ataque significativos, que cuando ocurren, pueden alertar a los analistas de seguridad [45].

2.7. Prelude

Prelude Hybrid IDS [46] es un Sistema de Gestión de Información de Seguridad universal sin agente (SIM, a.k.a SIEM) lanzado bajo los términos de la Licencia Pública General de GNU. Prelude recopila, normaliza, almacena y agrega registros de eventos de varios generadores de eventos. Además, convierte los eventos recopilados al formato estándar IDMEF. Prelude tiene los siguientes componentes principales [4]:

- *Prelude-Manager*: Prelude-Manager es el componente central de Prelude que puede conectarse tanto a los sensores de eventos como a otros administradores. Admite varios formatos de salida, como DB, Xmlmod, Textmod, Relaying y SMTP.

- *Libprelude*: Libprelude proporciona interfaces de programación de aplicaciones (API) que permiten a terceros comunicarse con Prelude-Manager.

- *LibpreludeDB:* LibpreludeDB es una biblioteca que proporciona una capa de abstracción para almacenar alertas IDMEF en la base de datos.

- *Prelude-LML:* Prelude-LML proporciona la capacidad de analizar diferentes tipos de registros de eventos.

- *Interfaz Prewikka:* La Interfaz Prewikka es la Interfaz Gráfica de Usuario (GUI) basada en la web para Prelude.

- *Prewikka-PFLogger:* El PFLogger recopila los registros de eventos del software PF de OpenBSD. Prelude recopila los tramos de eventos de los sensores y los convierte al formato IDMEF.

Este capítulo inició haciendo una introducción a los IDS y los diferentes tipos según su mecanismo de detección y su modo de operación, se analizaron comparativamente los dos tipos de IDS/IPS de estándar abierto más comunes como Snort y Suricata y varias soluciones propietarias como Darktrace, Cisco y FireEye. También se registraron diferentes formatos para el intercambio de eventos de seguridad, haciendo énfasis en el formato IDMEF por su uso en estándares abiertos. Es importante destacar la relación existente entre el análisis de vulnerabilidades y la correlación con las alarmas provenientes del IDS para lograr una mejor toma de decisiones y filtrar las alarmas de interés. Las herramientas de estándar abierto para hacer análisis de vulnerabilidades, como lo es Nessus, se sirven de referencias a bases de datos de vulnerabilidades como CVE y bugtraq con el objetivo de lograr un lenguaje común y estándar en el manejo de la información. Por último, se introduce Prelude SIEM como gestor de eventos de seguridad heterogéneos. Se destaca su capacidad para normalizar eventos en el formato IDMEF.
En el siguiente capítulo se hace un análisis de las diferentes estrategias para la gestión de alarmas y reducción de falsas alarmas propuestas en el estado del arte. A partir de ese análisis comparativo se extraen los elementos necesarios para la construcción de la estrategia híbrida de priorización.

CAPÍTULO 3

Consideraciones para el diseño de una estrategia híbrida para la priorización de alarmas de un IDS

Uno de los objetivos del proyecto es justamente definir una estrategia híbrida para la priorización de alarmas provenientes de IDS. Para ello, es necesario hacer una revisión de las estrategias que se han implementado, precisando sus resultados en términos de parámetros importantes como lo son la precisión, exactitud y sensibilidad con el objetivo de obtener una referencia común y extraer características útiles para la definición de la estrategia. En este capítulo se registran los antecedentes más relevantes en cuanto a los mecanismos y estrategias usadas para gestionar las alarmas provenientes de NIDS.

3.1. Análisis de antecedentes

El problema de minimizar el número de falsas alarmas y filtrar las alarmas de interés para el analista de seguridad ha sido ampliamente abordado en los últimos años [8]. Antes de ahondar en las técnicas de minimización de falsas alarmas es importante entender qué son las falsas alarmas y por qué se generan.
Un falso positivo es generado cuando el IDS dispara una alarma para un intento de ataque no exitoso y una falsa alarma es un conjunto de falsos positivos.
En [8] se mencionan dos razones importantes por las que un IDS puede generar falsas alarmas.

- La mayoría de IDS no solo detectan intrusos sino también el número de intentos de intrusión. Un intento no necesariamente puede conducir a un sistema comprometido. Estas alarmas son las que tiene que clasificar el analista de seguridad.
- En la mayoría de los casos, los IDS corren con un conjunto de reglas por defecto que no están personalizadas a la red local.

Debido al gran número de alarmas generadas por el IDS, los analistas de seguridad usan heurísticas propias de la experiencia en el ejercicio para priorizar las alarmas basados en la relevancia del ataque y en el impacto que pueda tener en la máquina objetivo.
Uno de los objetivos de algunas técnicas de minimización de falsas alarmas es abstraer esos métodos de razonamiento en algoritmos que conduzcan a aquellas alertas de interés.
En [47] se propone una clasificación de técnicas para la reducción de falsos positivos en dos categorías principales: *técnicas de detección y técnicas de procesamiento de alertas.*
Las técnicas de detección incluyen métodos que operan durante la fase de detección y las técnicas de procesamiento de alertas incluyen métodos que operan sobre las alertas producidas después de la fase de detección.
Debido a que la propuesta está contenida dentro de las técnicas de procesamiento de alertas, en este capítulo se ahonda en este tipo particular y más específicamente en las técnicas para procesar alertas obtenidas de sistemas de detección de intrusos basados en firmas.

En la literatura se encuentran una gran variedad de trabajos que clasifican las técnicas de gestión de alarmas. En particular, hay dos enfoques que consideran las técnicas de correlación de alarmas y las técnicas de minimización de falsas alarmas. En [48] por ejemplo, se hace una revisión y comparación de las diferentes técnicas de correlación de alertas provenientes de un IDS. Allí se define la

correlación de alertas como un proceso el cual recibe alertas provenientes de IDSs heterogéneos y reduce las falsas alertas, detecta patrones de ataque de alto nivel, incrementa el significado de los incidentes ocurridos, predice los estados futuros de los ataques y detecta las causas raíces de los ataques. En la figura 2 se muestran las técnicas de correlación de alarmas definidas en [48].
En [8] sin embargo, se hace una taxonomía de nueve técnicas usadas para minimizar las falsas alarmas generadas por IDS basados en firmas, y se define la correlación de alertas como una técnica macro de la cual se desprenden varias subcategorías. En la figura 3 se muestran las nueve técnicas consideradas en [8].

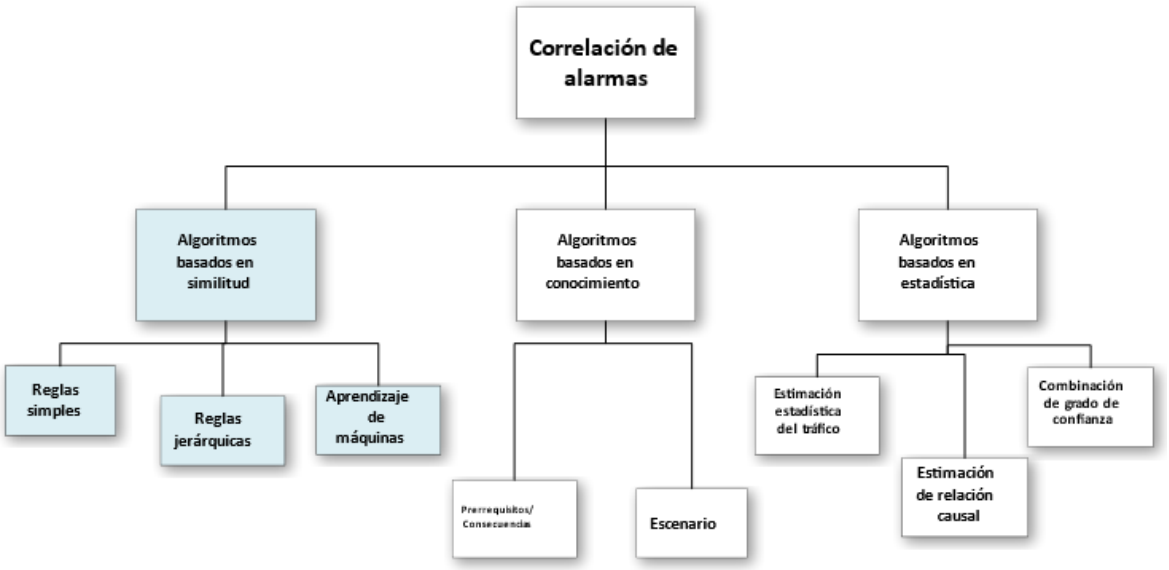

Figura 2. Algoritmos de correlación de alarmas [48].

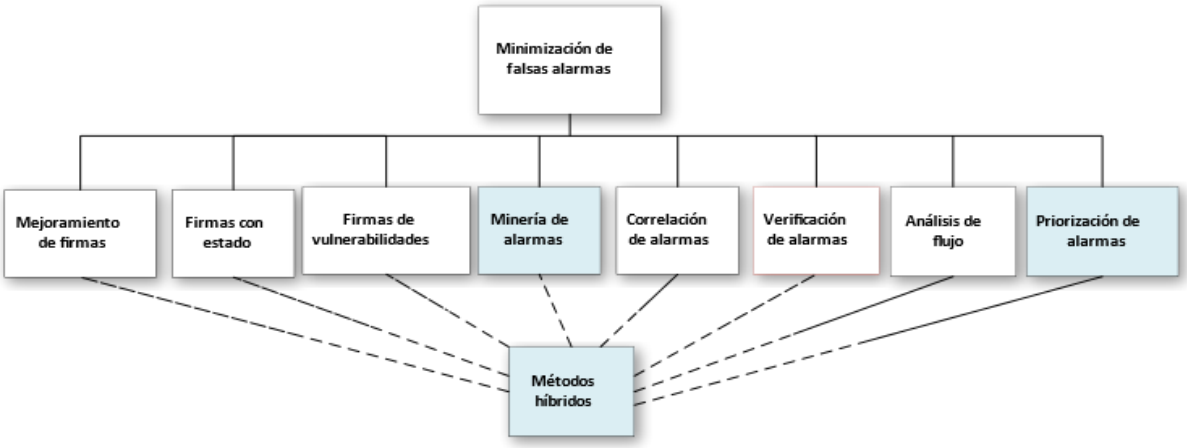

Figura 3. Técnicas de minimización de falsas alarmas propuestas en [8].

Luego de hacer una revisión extensa del estado del arte se concluye que en los dos trabajos ya citados se hace una vasta y completa clasificación y comparación tanto de las técnicas de correlación como de las de minimización de falsas alarmas en IDSs. En los siguientes párrafos se buscará trazar un paralelo entre los dos trabajos con el propósito de hacer un análisis más completo que abarque las

características y los factores de comparación que tiene en cuenta cada uno de ellos. La idea central es poder usar esta clasificación general como un instrumento para la justificación de la definición de la estrategia híbrida propuesta para la gestión de alarmas.

Antes de iniciar con la revisión de los trabajos consignados en el estado del arte y considerando que el objetivo principal de las técnicas para el procesamiento de alertas es el de reducir el volumen de alertas generadas por el IDS y filtrar las alarmas de interés para el analista de seguridad, es importante definir los parámetros de evaluación que harán posible la posterior comparación de cada estrategia. La efectividad de un IDS se evalúa según su habilidad de predicción, clasificando de manera correcta los eventos asociados a ataques reales o asociados al comportamiento normal de la red.

3.1.1. Parámetros de rendimiento asociados con la detección de intrusos

De acuerdo con la naturaleza real de un evento dado y la predicción de un IDS, en [47] se describen cuatro resultados posibles:

1. *Negativo Verdadero (TN*): son eventos normales que son exitosamente etiquetados como normales.
2. *Positivo Verdadero (TP):* son eventos relacionados con ataques que son exitosamente etiquetados como ataques.
3. *Falso positivo (FP):* son eventos normales siendo clasificados como ataques.
4. *Falso negativo (FN):* son eventos de ataques incorrectamente clasificados como eventos normales.

En la tabla 6 se describen los parámetros numéricos para evaluar la efectividad de un IDS:

Tabla 6. Parámetros de evaluación de un IDS y su equivalente numérico [47].

Parámetro de rendimiento	**Equivalente Numérico**
Tasa de Falsos Positivos (FPR)	$\frac{FP}{FP+TN}$
Tasa de Falsos Negativos (FNR)	$\frac{FN}{FN+TP}$
Tasa de Verdaderos Positivos (TPR)/Tasa de Detección	$\frac{TP}{TP+FN}$
Tasa de Verdaderos Negativos (TNR)	$\frac{TN}{FP+TN}$
Precisión	$\frac{TP}{TP+FP}$
Exactitud	$\frac{TN+TP}{TP+TN+FP+FN}$
Sensibilidad	$\frac{TP}{TP+FN}$

La exactitud se refiere a la proporción de eventos clasificados de forma exacta, en nuestro caso TN y TP sobre el total de eventos. La precisión da la medida del número de eventos críticos o relacionados con ataques sobre el total de alarmas clasificadas como ataques. Por su parte, la sensibilidad se refiere a la proporción de alarmas clasificadas como no críticas.

Para optimizar el rendimiento de un IDS se debe minimizar las tasas de FP y FN, maximizando la exactitud y las tasas de TN y TP. Con la sola reducción de los FP no es suficiente. Algunas técnicas de reducción de falsos positivos causan baja exactitud debido a que usan estrategias de sobre-generalización de reglas.

3.1.2. Parámetros de desempeño asociados con los algoritmos de detección y/o clasificación de alarmas

Adicional a los parámetros de desempeño asociados con la detección o clasificación de alarmas, es importante entender los factores de desempeño asociados con los algoritmos de detección o clasificación de alarmas. En [48] se mencionan algunos de interés:

Capacidad del algoritmo: la capacidad del algoritmo se entiende como la habilidad que este tiene para verificar las alertas, agrupar alertas similares (*clustering*), detectar secuencias de ataques y reducir las alertas repetitivas que no son de interés.

Exactitud del algoritmo: la exactitud del algoritmo da razón del número de errores que comete el algoritmo en el proceso de resumir los estados del sistema y en reducir el volumen de alertas a las alertas de interés.

Potencia computacional del algoritmo: la potencia computacional se relaciona con la cantidad de operaciones que tiene que ejecutar el motor de correlación evaluando el uso de memoria y la potencia de procesamiento que requiere cada algoritmo.

Flexibilidad y extensibilidad del algoritmo: este factor de evaluación da cuenta de qué tan cambiante, localizable y adaptable a las nuevas condiciones es el rendimiento del algoritmo.

En [49] se hace una clasificación de tres tipos de algoritmos principales: *algoritmos basados en similitud, algoritmos basados en conocimiento y algoritmos basados en estadísticas.* Por razones de la metodología propuesta para el desarrollo del proyecto, no se consideran los algoritmos basados en estadísticas para la implementación de la estrategia, mas sí se tienen en cuenta en el análisis de antecedentes y en la comparación para la elección de la arquitectura de la estrategia.

3.1.3. Algoritmos basados en similitud y basados en conocimiento

El objetivo de los *algoritmos basados en similitud* es el de agrupar alertas similares en el tiempo. La ventaja más importante de estos algoritmos es que no hay necesidad de una definición precisa de los tipos de ataque. En lugar de ello, la correlación se puede establecer sólo con la definición de factores de similitud para diferentes características de las alertas. En esta categoría se definen tres subcategorías: *similitud establecida mediante reglas simples, reglas jerárquicas, y factores generados automáticamente mediante aprendizaje de máquinas.*

En la *similitud basada en reglas simples* el conocimiento que se requiere es el de estructuras de reglas y funciones necesarias para chequear la similitud. Dentro de los algoritmos que establecen el grado de *similitud usando reglas jerárquicas* se encuentran aquellos que se diseñan para detectar las causas raíces de las alarmas generadas. Con relación a los *factores de similitud generados mediante*

aprendizaje de máquinas, este tipo de algoritmos requiere un conjunto enorme y completo de ejemplos de entrenamiento para crear los árboles de decisión y realizar la correlación. Algunos algoritmos dentro de esta subcategoría realizan agrupamiento (*clustering*) de alarmas basado en las estadísticas del *cluster* por lo que el conocimiento previo que se requiere es el del conjunto de alertas ya examinadas. Por otro lado, hay otro tipo de algoritmos que tienen la capacidad de entrenamiento en línea (*online*), tienen un alto grado de flexibilidad y la toma de decisiones está basada principalmente en información relacionada con alertas de tipo similar en rangos de tiempo cercanos al arribo de nuevas alertas.

La categoría que en [48] se describe como algoritmos basados en similitud, en [8] se conoce como algoritmos basados en conocimiento. Este tipo de algoritmos usa una extensa base de conocimiento acerca del sistema que está siendo monitoreado. Este conocimiento puede ser estático o dinámico. La base de conocimiento está representada por el tipo de sistema operativo soportado por los equipos dentro de la red objetivo, tipo de aplicaciones corriendo y algunas veces las vulnerabilidades conocidas. Este tipo de algoritmos agrupa las alarmas con base en métricas de similitud. Para ello, usa un conjunto de características identificadas para la medida de distancia y la posterior agrupación. Mediante la fusión de múltiples alarmas del mismo tipo, el administrador de seguridad queda con información condensada que da una visión más amplia acerca de la situación de ataque.

Con relación a los algoritmos basados en conocimiento, en [49] se propone un método que captura los datos que definen las políticas de seguridad y las alarmas generadas por múltiples IDS. Se discute una arquitectura llamada SCYLLARUS la cual usa un componente llamado Modelo de Referencia de Intrusión (IRM por sus siglas en inglés). El IRM tiene información dinámica y estática en forma de una base de datos de eventos, configuración del hardware y software de la red, así como una base de datos de objetivos de seguridad.
El primer componente del IRM se conoce como preprocesador de *cluster* que ensambla un conjunto de alarmas de IDS relacionadas. El siguiente componente se conoce como un descriptor de acceso (*Accessor*) que examina un conjunto de eventos de las bases de datos y encuentra una relación entre las alarmas y los eventos para determinar alarmas plausibles. Una vez el conjunto de alarmas plausibles es determinado, se establece el impacto de estas alarmas con base en la evaluación de esas alarmas contra las políticas de la red considerada.
Aludiendo a la correlación basada en conocimiento, Sourcefire [50] tiene un producto llamado Sourcefire IPS el cual posee un motor inteligente que puede elegir y habilitar automáticamente reglas de Snort requeridas según los activos de la red.
En [51] se describe un modelo basado en relaciones lógicas entre los hosts, vulnerabilidades y otros eventos complementarios. Este modelo proporciona un sistema para correlacionar la información de salida del escáner de vulnerabilidades con las alarmas del IDS.

Continuando con el análisis, las subcategorías que en [48] se describen como algoritmos de aprendizaje de máquinas y algoritmos basados en reglas jerárquicas, que a su vez están contenidos en la categoría de algoritmos basados en similitud, en [8] están contenidas dentro de una categoría de técnicas de *minería de alarmas*.

3.1.4. Técnicas de minería de alarmas

Las técnicas de minería de alarmas usan atributos como direcciones IP, número de puertos, información de protocolos y demás para minar un conjunto de alarmas y clasificarlas en falsos positivos o en alarmas que conducen a hosts comprometidos. El concepto de minar se refiere al proceso que intenta descubrir patrones en grandes volúmenes de datos usando métodos de inteligencia artificial, aprendizaje automático, estadística y demás.

Las características aprendidas durante la etapa de minado son usadas para clasificar futuras alarmas. Las técnicas de minería de alarmas se pueden clasificar en *clustering*, *clasificación*, *modelos basados en redes neuronales* y en *minería de patrones frecuentes* [8].
Dado que estas técnicas de minería de alarmas usan atributos de las alarmas para efectuar su agrupación o clasificación basado en medidas de distancia o patrones de frecuencia, no es impreciso decir que estas técnicas están contenidas dentro de la categoría de correlación basada en similitud.

En lo que sigue, se abordan algunos trabajos categorizados dentro de las técnicas de minería de alarmas haciendo énfasis en los parámetros de rendimiento como reducción del volumen de alarmas, precisión, sensibilidad y capacidad de operación en tiempo real.

3.1.4.1. Clustering de alarmas

En esta técnica se toman las alarmas generadas por el IDS y se crean grupos (*clusters*) de tipo similar. Luego, a estos grupos se les asigna un significado de falsas alarmas y alarmas verdaderas, es decir, todas las alarmas en el grupo son ya sean falsas alarmas o alarmas verdaderas.
En una serie de investigaciones realizadas por Julisch et al. [52-55] se afirma que, aprendiendo los patrones de las falsas alarmas y las causas asociadas al origen de estas, las alarmas futuras se pueden clasificar como falsas o verdaderas. Para los experimentos se usaron dos algoritmos diferentes de minería de datos, a saber, minería de episodios y algoritmos de *clustering*. Usando minería de episodios se descubren un conjunto de episodios y se filtran aquellos que corresponden al comportamiento normal del sistema. Los experimentos llevados a cabo en redes reales revelan reducciones en la carga de alarmas del 75% [52] y 87% [54].

En [56,57,58] se reportan trabajos donde usan análisis de causa-raíz para descubrir las causas que provocan que el IDS dispare falsas alarmas. Esta causa-raíz hace que el IDS dispare alarmas que generalmente tienen características similares. Los autores desarrollaron una técnica de *clustering* para agrupar las alarmas del IDS y producir *clusters*. Cada *cluster* es modelado por una alarma generalizada. A su vez, cada alarma generalizada que está relacionada con una causa-raíz se convierte en filtros para reducir la carga de alarmas futuras. Se logran reducciones del volumen de alertas generadas del 82% [56], 74% [57], 93% en una red real y 70% usando el conjunto de datos DARPA 1999 [58].

En [59] se describe un método de *clustering* de alarmas en el cual las alarmas se clasifican en clases predefinidas. Los grupos de interés se forman inicialmente mediante la creación de listas vacías. Las alarmas son añadidas a estos grupos de manera incremental. Cada alarma es comparada al representativo del grupo, llamado meta alarma, usando una función de distancia personalizada. La alarma es asignada a uno de los grupos cuya distancia con la alarma sea mínima. En [60] usan K-medias para generar los grupos.
En [61] se presenta una técnica de minería de alarmas basada en mapas auto-organizados con crecimiento jerárquico (GHSOM por sus siglas en inglés) que ajusta su arquitectura durante el proceso de entrenamiento no supervisado de acuerdo a las características de las alarmas de entrada. Con relación a los resultados obtenidos, se logró una reducción de los falsos positivos del 15% al 4.7% y de los falsos negativos del 16% al 4%.

En [62] se propone un marco de referencia para el procesamiento de alertas que reduce la cantidad de alertas a ser procesadas y crea escenarios de ataque más significativos para el post análisis. Los autores usan una técnica de *clustering* donde a cada alerta se le extraen tres atributos (IP destino, tipo de ataque o tipo de firma y marca de tiempo) y se calcula un valor Hash para cada alerta en formato MD5. Se comenta que comparar números de la tabla Hash es mucho más rápido que comparar texto.

Con relación a los resultados obtenidos, se lograron reducciones de hasta un 86.9% en el volumen de alertas generadas. Se usó el conjunto de datos DARPA.

3.1.4.2. Clasificación de Alarmas

Este método asume un conjunto de alarmas etiquetadas para entrenar el algoritmo de clasificación. El proceso de etiquetado es llevado a cabo por un experto que según su experiencia etiqueta las alarmas como falsas o verdaderas. Una vez el algoritmo está entrenado, puede ser usado para clasificar alarmas futuras.
En [63] los autores proponen un modelo híbrido en el cual una parte comprende la minería de datos de un histórico de alarmas pasando el conocimiento aprendido a un analista experto y la otra parte sintoniza el motor de firmas del IDS para reflejar el comportamiento aprendido. Este esquema usa varias entidades de red para construir el clasificador ALAC, las cuales son representadas como una topología en árbol parecido a un árbol de decisión. Las alarmas son agrupadas según la similitud de atributos como dirección IP y números de puerto, después de atravesar el árbol. Los resultados en términos de reducción del volumen de alertas fueron del 50%. En [8] comentan que esta propuesta contiene una limitación en términos de utilidad práctica puesto que se involucra el componente humano en el ciclo.

En [64] los autores proponen un enfoque supervisado de clasificación de alarmas del IDS para la minimización de falsas alarmas. La técnica es llamada Aprendiz Adaptativo para la Clasificación de Alarmas (ALAC por sus siglas en inglés) en la cual se involucra al componente humano en el ciclo. La reducción de falsas alarmas fue de 30%.
En [65] el autor propone un método de clasificación en tiempo real basado en minería de datos para distinguir alertas importantes de los falsos positivos que ocurren con frecuencia y los eventos de baja importancia. El autor asegura que contrario a los enfoques convencionales basados en minería de datos, el método es totalmente automatizado y permite ajustarse a cambios en el ambiente sin intervención humana. Los resultados en términos de precisión fueron del 97.02% y en sensibilidad del 99.96%. Luego, el mismo autor extiende su trabajo previo en [66] y presenta un método de clasificación de alertas en tiempo real no supervisado el cual está basado en la minería de conjuntos de datos frecuentes y técnicas de *clustering*. Los resultados en términos de precisión fueron del 98.65% y se midió una sensibilidad del 99.96%. El algoritmo opera en tiempo real y los datos de entrenamiento son actualizados periódicamente.

En [67] se describe un clasificador de alertas adaptativo basado en minería de patrones. El clasificador se construye usando el algoritmo a priori basado en poda recursiva. El clasificador de alertas, ayuda al operador de seguridad mediante la clasificación de alertas en falsas o positivas aprendiendo nuevos patrones de alarmas de manera adaptativa mediante la realimentación de los operadores. Los resultados de la implementación de este método muestran reducciones de hasta el 36% del volumen de alertas generadas. El conjunto de datos usado fue el DARPA-1999.

3.1.4.3. Minería de patrones frecuentes

La minería de patrones frecuentes es una técnica para identificar conjuntos de elementos frecuentes en una base de datos de transacciones dada. En el contexto de IDS, las alarmas generadas por el IDS son transacciones y las combinaciones de alarmas frecuentes indican una secuencia que se está repitiendo. Estos patrones repitiéndose indican las acciones que un intruso ha intentado antes de penetrar en el host objetivo.
En [68] se propone un esquema de clasificación de alertas en tiempo real basado en patrones frecuentes estructurados. Este esquema está constituido por un componente llamado agregador que transforma las alarmas sin procesar en gráfos, luego de analizar la relación de conectividad entre ellas.

Esta entrada es seguida por una estructura que extrae patrones vistos con frecuencia en el pasado reciente y construye un árbol llamado árbol de patrones frecuentes. La salida de este componente es un modelo de correlación llamado modelo en ejecución. Este modelo es dinámico y puede cambiar con el tiempo. Con este método se logró una tasa de reducción de alarmas del 96%. El conjunto de datos usado fue el DARPA 2000. La correlación de alertas se hace en tiempo real.

En [69] se realiza un análisis secuencial de las alarmas para encontrar las alarmas críticas. Cada tiempo de ocurrencia de una alarma es usado para ordenar las alarmas en una secuencia. Se elije una ventana de tamaño *w* y todas las alarmas contenidas en esta ventana son denominadas episodios. Una vez los episodios son descubiertos, se usa una función de clasificación para clasificar los episodios y aquellos con mayor *rating* son elegidos como los episodios críticos. Con este método se logró una tasa de reducción de alarmas del 90%. El conjunto de datos usados fue el LL DDoS 1.0.

Dentro de la segunda categoría principal (algoritmos basados en conocimiento) que se establece en [48], los autores presentan dos subcategorías, a saber, *algoritmos de prerrequisitos y consecuencias* y *algoritmos para el descubrimiento de escenarios de ataque*.
En la primera categoría se requiere de un conocimiento previo para determinar los prerrequisitos y todos los resultados de incidentes existentes. En este tipo de algoritmos las alertas son modeladas usando relaciones causales. A partir de este conocimiento se puede construir un grafo de posibles alertas y las relaciones entre ellas, para así, mediante el uso de herramientas apropiadas reducir la cantidad de información mostrada al usuario. Dado que este tipo de algoritmos no usa ninguna información pre-asumida adicional al conocimiento por defecto del ambiente, son muy flexibles y extensibles, a parte que su comportamiento cambia en tiempo real con cualquier cambio en el conocimiento del ambiente.
En cuanto a los algoritmos de escenario, la aplicación principal de este conjunto de algoritmos yace en la detección de ataques multi-etapa y en la confianza en los escenarios existentes para este tipo de ataques.

3.1.5. Algoritmos basados en estadísticas

Los *algoritmos basados en estadística* guardan relaciones causales entre diferentes incidentes y analizan su frecuencia de ocurrencia en el periodo de educación del sistema usando análisis estadístico de datos previos y luego generando las etapas del ataque.
Esta categoría también está dividida en tres subcategorías. La primera subcategoría es la de *estimación estadística del tráfico***.** El objetivo de este tipo de algoritmos es el de detectar alertas que están repetidas con cierta regularidad y encontrar su patrón de repetición. Ciertos algoritmos [48] crean un modelo estadístico del tráfico de la red, prediciéndolo y removiendo los casos predecibles. Una categoría importante de este tipo de alertas contiene las alertas que ocurren periódicamente debido a los ajustes erróneos en la red.
Una característica a resaltar de este tipo de algoritmos es que los modelos estadísticos que emplean en los trabajos referenciados incluyen entrenamiento online, lo que da pie a tener un registro de las condiciones actuales de la red y la flexibilidad basada en el seguimiento a cambios nuevos en la misma.
Los algoritmos basados en reglas están contenidos dentro de esta subcategoría. Este tipo de reglas se usan para detectar alertas que usualmente ocurren juntas. El entrenamiento es llevado a cabo offline, pero se pueden actualizar parámetros del modelo en tiempo de ejecución y optimizarlo de acuerdo a nuevos datos.

En la segunda subcategoría están contenidos los *algoritmos basados en la estimación de relación causal.* El propósito de esta subcategoría es el de encontrar secuencias de alertas o asociaciones de patrones dominantes y usar estos patrones para detectar casos falsos o combinaciones impropias. En

[70] usan técnicas de detección de anomalías sobre el conjunto de alarmas producidas por varios sensores detectores de intrusos para reducir el número de falsas alertas. En concreto, los autores establecen una línea base del comportamiento de un sensor (IDS) y detectan desviaciones de esta línea base. Los autores muestran que las desviaciones de este perfil creado tienen una alta probabilidad de ser ataques reales. Se evalúan heurísticas Markovianas contra el algoritmo de compresión Lempel-Ziv y se muestra que usando esta estrategia se detectan todos los ataques que fueron manualmente identificados por el personal de seguridad además de detectar aquellos ataques que fueron pasados por alto durante la evaluación manual. En términos numéricos, la heurística reportó una reducción del 99% del total de las alarmas usando cadenas de Markov. Usando el modelo oculto de Markov (Hidden Markov Models) se logró una reducción del 95% y con el algoritmo de Lempel-Ziv del 88%. En los tres casos se detectó el 100% de los ataques conocidos.

La tercera subcategoría se conoce como *combinación de grado de fiabilidad*. En este tipo de algoritmos, se propone cambiar la fiabilidad de las alertas basado en repeticiones de alertas equivalentes. El objetivo es cambiar la prioridad de determinada alerta basado en la información de otros recursos de la red. La subcategoría de ataques basados en escenario contenida en la categoría de algoritmos basados en conocimiento, en [8] se conoce como correlación multi-etapa. Este tipo de algoritmos asume que hay una secuencia de acciones seguidas por el atacante antes de irrumpir en el sistema.
Ahora, en [8] se describe una categoría de correlación basada en relaciones causales. Se menciona que el objetivo del descubrimiento de relaciones causales es el de chequear e identificar relaciones causales entre las variables bajo estudio. Dadas dos o más variables aleatorias esta correlación identifica cómo está relacionada cada variable con la otra. Esta definición es bastante precisa con relación a la que usan en [48] para la categoría de algoritmos basados en estadísticas.
Teniendo en cuenta las relaciones establecidas entre cada uno de los enfoques, las técnicas de priorización [8] donde se requiere de la definición de una fórmula de puntaje basado en un conocimiento previo de la red, se relacionan fundamentalmente con la correlación basada en similitud.

3.1.6. Técnicas de Priorización

Las técnicas de priorización califican las alarmas con base en una post evaluación. Dado un conjunto de alarmas, al analista de seguridad recibe una lista con las alarmas priorizadas. Para generar este valor se pueden considerar aspectos de la topología de red, historial del IDS, lugar de ubicación del IDS en la red, etc.
En [5] se propone una técnica para clasificar las alarmas llamada M-Correlator. En primer lugar, esta técnica usa alarmas de diferentes sistemas de seguridad como IDS y firewalls, configuración de la red y parámetros definidos por usuario como criticidad de las aplicaciones, cantidad de interés en un tipo de ataque, etc. Toda esta información es correlacionada para generar una clasificación de alarmas particulares.
En [6] se propone una técnica de priorización de alarmas basada en lógica difusa que genera un score de prioridad considerando varias métricas. Se considera una métrica de aplicabilidad que captura la relevancia de un ataque particular considerando el contexto de la red. Importancia, que captura qué tan importante es la entidad objetivo, estado del sensor que toma en cuenta si el IDS del que proviene la alarma está bien configurado o no, severidad que captura qué tan severo es el ataque y por último se tiene una métrica de historia que captura el histórico del IDS. Con relación a los resultados obtenidos hay que decir que se logró una reducción del 92.57% del total de alertas con agrupamiento y del 64.83% sin agrupamiento. El conjunto de datos usados fue el DARPA 2000 LLDOS 1.0.

En [71] se presenta una técnica que los autores categorizan dentro de los esquemas de verificación de alertas. El enfoque de verificación de alertas que se presenta está basado en la compresión difusa

multinivel. Los autores aseguran que correlacionar la información de la topología con las alarmas no es adecuado. Se menciona que la correlación de información de topología incompleta con las alarmas puede inducir errores en la decisión final. Para evitar esto computan una matriz creada a partir de la correlación de atributos de las alarmas e información de la topología. Mediante algunas operaciones matemáticas derivan un score de relevancia con ayuda de esta matriz. Los autores compararon el enfoque propuesto con Snort usando tres métricas de desempeño: tasa de detección, tasas de falsas alertas y tasa de alertas irrelevantes. En términos de tasa de detección usando Snort registraron un 65.6%, con la técnica propuesta registraron el mismo valor. Con Snort registraron un 93.8% de falsas alarmas y un 86% de alarmas irrelevantes. Mediante el uso de la técnica propuesta lograron reducir la tasa de falsas alarmas al 33.4% y la proporción de alarmas irrelevantes al 15.3%.

En [72] los autores proponen una técnica articulada de tres componentes (1) alertas vecinas relacionadas (NRA), (2) un componente de frecuencia de alerta alta (HAF) y (3) un componente de falsos positivos usuales (UFP). El conjunto de alarmas producido por el IDS se lleva a cada componente y cada uno de estos componentes genera un score para cada alarma indicando la probabilidad de que la alarma sea una falsa alarma. Finalmente, estos scores se combinan para generar un score final a partir del cual se deduce si la alarma es una falsa alarma o una alarma verdadera. Con la implementación de esta técnica se percibió una reducción del 28.83% del número de alertas, el porcentaje de ataques detectados se redujo en un 4.16%, el número de falsos positivo se redujo en un 73.98%.

En [73] se presentan los resultados de una técnica para la priorización de alertas provenientes de IDSs. Usan el Marco de Utilidad de Lógica Difusa (FLUF por sus siglas en inglés) el cuál fue desarrollado con una base de reglas diseñadas con los activos de red críticos y la misión de negocio en mente.
El objetivo principal de este enfoque es el de reducir el tiempo entre el que un analista observa una alerta en particular y el tiempo que tarda en iniciar la investigación de la alerta, con la asignación de la prioridad basada en la criticidad de las alertas. Los resultados obtenidos empíricamente en un experimento en el que usaron el conjunto de datos CDX2009 los cuales provienen de un Ejercicio de Ciber Defensa (CDX) patrocinado por la Agencia de Seguridad Nacional (NSA), demuestran una reducción importante en el tiempo de descubrimiento de una alerta cuando se usa la métrica del rango de prioridad de una alerta (APR) FLUF para ayudar a los analistas a enfocarse en las alertas más críticas.

3.1.7. Técnicas Híbridas

Las técnicas híbridas no son más que una combinación de cualquiera de las técnicas previamente abordadas.
En [74] se propone un proceso de dos etapas basado en minería de datos y optimización, el cual recibe como entrada las alarmas generadas por múltiples IDS. En la primera etapa, por cada IDS se agrupa el conjunto de alertas elementales para crear un conjunto de meta-alertas. Como paso posterior, se remueven los falsos positivos del conjunto de meta alertas usando un problema de optimización binario. En la segunda etapa, se descartan las meta-alarmas generadas por cualquier IDS y solo aquellas perdidas por uno, dos o más de ellos se dejan. Este conjunto se denomina el conjunto de falsos negativos potenciales. En este nivel se realiza una fusión de alertas para evitar la redundancia entre las meta-alertas recogidas desde múltiples IDSs. Como paso final, se propone un algoritmo de clasificación binaria para clasificar los falsos negativos potenciales ya sean como ataques reales o no. En cuanto a los resultados experimentales se obtuvo lo siguiente:
Se usaron tres métodos diferentes para la reducción de falsos positivos con dos conjuntos de pruebas diferentes. Aquí sólo referenciamos aquellos resultados obtenidos con el conjunto de prueba 1. Usando el método de BOP (problema de optimización binario) lograron una tasa de positivos verdaderos (TPR) de 95% y una tasa de negativos verdaderos (TNR) de 94%. Con el esquema de

votación ponderada basada en credibilidad (CWV) consiguieron un TPR de 68% y un TNR de 65.3%. Usando mapas autoorganizados con k-medias (SOM-KM) TPR 78% y TNR 75%. Es evidente pues que usando BOP se lograron mejores resultados.
Con relación a los métodos para reducir los falsos negativos, usaron un algoritmo de clasificación binaria (BCA) con el que lograron una TPR de 93.5% y una TNR de 94.5%. Con el algoritmo de voto mayoritario (MV) lograron una TPR de 54.8% y una TNR de 58%. Finalmente, con el algoritmo de CWV lograron una TPR de 72% y una TNR de 79%. Como trabajo futuro se propuso analizar los resultados de la estrategia propuesta usando datos tráfico de red real. Usaron el conjunto de datos DARPA 1999.

En [75] se propone un enfoque híbrido que involucra tanto la técnica de filtrado como la de clasificación. El filtro de falsas alarmas incluye:

- Un perfil de amenazas dinámico representando las vulnerabilidades presentes en la red. Este perfil de amenaza es construido periódicamente para obtener una visión más consistente de las amenazas de la red local. El perfil puede ser generado manteniendo una base de datos de todas las vulnerabilidades conocidas que son publicadas regularmente. Las bases de datos de Bugtraq y CVE son las mejores fuentes para esto. Además, se usan escáneres de vulnerabilidades como Nessus, Nmap o Retina para generar los perfiles de amenaza.
- Un motor de correlación basado en redes neuronales para filtrar las alarmas. Este motor correlaciona el perfil de amenaza con las alarmas y filtra las falsas alarmas. Inicialmente se genera un conjunto de alarmas provenientes del IDS usando programas de ataque y luego estas alarmas son etiquetadas. Estas alarmas etiquetadas son usadas para el entrenamiento de la red neuronal.

Con relación a los resultados obtenidos, se computaron las métricas de precisión y tasa de detección para seis clases de ataques: Telnet, DoS, FTP, SQL, MySQL. Haciendo un promedio de los resultados dentro cada clase, se obtiene una precisión promedio de 97% y una tasa de detección de 95.46%.

En [76] se usa información contextual basada en la configuración del objetivo del ataque (sistemas operativos y aplicaciones) para la identificación de alarmas no críticas. Se demuestra que la información de sistemas operativos y aplicaciones son piezas de información complementarias debido a que se obtienen mejores resultados en términos de tasa de reducción de las alarmas. Los resultados en términos de precisión y sensibilidad fueron del 99.6% y del 73.1%.
En [77] se describe un sistema de clasificación de alertas automático de dos fases para asistir al analista humano en la tarea de identificar los falsos positivos. En la primera fase, las alertas colectadas de uno o más sensores son normalizadas y las alertas similares se agrupan para formar una meta alerta. Estas meta-alertas son verificadas pasivamente con una base de datos de activos de red para encontrar alertas irrelevantes. Adicionalmente, se realiza una generalización de alertas para análisis de causas raíces y por lo tanto reducir los falsos positivos con interacción humana. En la segunda fase las alertas reducidas son etiquetadas y pasadas a un clasificador de alertas el cuál usa técnicas de aprendizaje de máquinas para construir las reglas de clasificación. Es de notar que el proceso de etiquetado de alertas para entrenar el algoritmo se automatiza usando el mecanismo de verificación, evitando así los tiempos de baja del sistema programados periódicamente para reentrenar el clasificador. Este sistema se testeó usando varios esquemas de clasificación. En general, en términos de precisión y sensibilidad se obtuvieron buenos resultados tanto en un ambiente real como también usando el conjunto de datos DARPA 1999. Sin embargo, comparando la FNR, los mejores resultados se lograron usando Ripper como el aprendiz de regla por defecto. La precisión para el conjunto de datos reales fue de 99.1% y la sensibilidad de 99.1%. Usando el conjunto de datos DARPA 1999 lograron una precisión de 95.4% y una sensibilidad de 99.6%.

En las tablas 7 y 8 se registran de manera sucinta los resultados obtenidos en cada trabajo referenciado en términos de los parámetros de evaluación ya mencionados.

Tabla 7. Registro de los resultados de las diferentes estrategias clasificadas según el enfoque propuesto en [8] en términos de los parámetros de rendimiento mencionados.

	Técnica de minimización de FA	Precisión	Sensibilidad	Tasa de reducción de alarmas
Minería de Alarmas	**Clustering de alarmas**	FP del 15% al 4.7% [61] -	FN del 16% al 4% [61] -	75% [52], 87% [54], 82% [56], 74% [57], 93% [58], 86.9% [62]
	Clasificación de alarmas	30% en reducción de FA [64], 97.02% [65], 98.65% [66]	99.96% [65], 99.96% [66]	50% [63], 36% [67]
	Minería de patrones frecuentes	-	-	96% [68], 90% [69]
	Algoritmos basados en estimación de relación causal	-	-	99%, 95%, 88% [70]
	Técnicas de priorización	33,4% FA [71], FPR en 63.42% [72]	-	92.57% (sin/a), 64.83% (con/a) [6], 15.3% [71], 28.83% [72]
	Técnicas híbridas	97%[75], 99.6% [76], **97.25% [77]**	73.1%[76], **99.3%[77]**	TPR 95%, 93.5% [74], TNR 94%, 94.5% [74], 95.46% [75]

3.2. Acerca de la operación en tiempo real

Si bien en algunos trabajos referenciados en la sección anterior se mencionó que el procesamiento de las alarmas se hacía en tiempo real, hasta aquí no se ha hecho ningún análisis acerca de las condiciones necesarias para habilitar esta opción. La pregunta que surge es ¿Qué es operación en tiempo real? ¿Quizá se relacione a tiempos de respuesta del orden de milisegundos en el procesamiento de una alarma? En realidad, para hablar de operación en tiempo real es necesario conocer la cantidad máxima de alarmas generadas en un periodo de tiempo y cuáles son los recursos de procesamiento con los que se cuenta por parte de los nodos IDS y de las consolas de gestión para atender esta demanda de alarmas [78]. Por defecto, los nodos IDS tienen una restricción de capacidad de procesamiento propio del hardware. Si se piensa en los SIEM, e independiente de si el esquema de operación es distribuido

o centralizado, esta restricción propia de los equipos de hardware persistirá. Para superar esa restricción propia de los equipos es necesario disponer de estrategias que una vez desbordados los recursos del sistema permita la agregación de nodos y distribuya de una manera inteligente la carga de trabajo según la capacidad de procesamiento de cada uno, cubriendo así la demanda total.

En respuesta a lo anterior, una de las soluciones que surgen es la paralelización de los datos a procesar, técnica usada por defecto para la gestión de grandes volúmenes de datos. En la paralelización de datos, estos son distribuidos a través de un clúster de ordenadores y la misma tarea se ejecuta en paralelo en cada nodo del clúster.
Las tecnologías Big Data se pueden categorizar en dos clases principales: procesamiento de flujo (*stream processing*) y procesamiento por lotes (*batch processing*) [78]. El procesamiento de flujos se realiza justamente sobre el flujo de datos mientras que el procesamiento por lotes se realiza sobre los datos almacenados. Una tecnología de Big Data que usa procesamiento por lotes es Hadoop. Para realizar el procesamiento por flujos se tiene *Spark Streaming*, *Apache Flink*, *Apache Kafka* etc [79]. Con este tipo de tecnologías se pueden analizar grandes conjuntos de datos heterogéneos a alta velocidad, transformando el análisis de la seguridad al capturar gran cantidad de datos de numerosas fuentes como bases de datos de vulnerabilidades, realizar análisis profundos sobre los datos, logrando análisis en tiempo real de los flujos de datos, entre otros.

La detección y predicción de intrusos en la red debería realizarse si no en tiempo real cercano a tiempo real por lo que las tecnologías Big Data surgen como candidatos a solución. Para refrendar lo anterior, en [79] se introduce una técnica para la correlación de alarmas usando Hadoop. La arquitectura propuesta se compone de un módulo de agregación, verificación y correlación. Se demuestra que todos los componentes de la arquitectura son escalables en varios nodos. Adicionalmente, en [80] se presenta una aplicación que combina el procesamiento de flujos, el procesamiento por lotes y técnicas de aprendizaje de máquinas para verificar las falsas alarmas en un ambiente industrial. Los autores usan un conjunto de datos de 350K alarmas reales de sensores implementados en producción, evalúan 4 algoritmos de aprendizaje automático (random forest, máquinas de soporte vectorial (SVM por sus siglas en inglés), regresión logística y redes neuronales profundas (DNN por sus siglas en inglés)) y obtienen una clasificación de alarmas con una precisión de más del 90% usando el algoritmo Random Forest y redes neuronales profundas.
Si bien en la referencia anterior se verifican alarmas provenientes de sensores industriales, no IDS, se podrían mapear exactamente la misma arquitectura y usar las mismas herramientas.

Con lo anterior en mente, es tiempo de mostrar los resultados del análisis comparativo realizado en la sección anterior juzgando los factores de rendimiento asociado con los algoritmos, en especial la característica de paralelización de los algoritmos (tabla 8).
La asociación tanto de las técnicas de minimización de alarmas como las de correlación de alarmas se establece según tres categorías de alto nivel: algoritmos basados en similitud, basados en estadísticas y basados en conocimiento. La tabla original se extrae de [48], y se le adiciona la columna izquierda (según [8]). El objetivo primario es el de establecer la relación entre la correlación basada en conocimiento, la minería de alarmas y la priorización de alarmas con los valores asignados a los factores de rendimiento para los algoritmos de correlación basados en reglas simples y reglas jerárquicas. A partir de la relación establecida, según los trabajos en las referencias analizadas y según las características de los algoritmos, se concluye que los algoritmos de correlación basados en similitud mediante reglas simples y las técnicas de priorización son altamente paralelizables y por tanto escalables en ambientes de grandes volúmenes de datos.

Tabla 8. Asociación y clasificación de las técnicas de gestión de alarmas según los enfoques propuestos en [8] y [48]. B.C. quiere decir basados en conocimiento.

	H	Alto							
	A	Promedio							
	L	Bajo							
Según [8]			**Según [49]**	**Exactitud**	**Flexibilidad**	**Extensibilidad**	**Memoria requerida**	**Potencia computacional**	**Paralelización**
Correlación basada en conocimiento		**Basados en similitud**	Reglas simples	A	H	H	A	A	H
Minería de alarmas			Reglas jeráquicas	A	H	H	A	A	H
Priorización de alarmas			Máquinas de aprendizaje (Árboles de decisión)	A	A	A	A	A	A
			Máquinas de aprendizaje (Re-creación)	A	A	A	L	H	H
			Máquinas de aprendizaje (Verificación)	A	A	A	H	H	L
Correlación basada en grafos de ataques		**B. C.**	Prerrequisitos/Consecuencias	H	H	H	H	L	L
Correlación multi-etapa			Escenario	H	H	H	A	A	L
Correlación basada en relaciones causales		**Basados en estadísticas**	Estimación de tráfico estadístico (ETE)	A	H	A	L	H	H
			ETE (Reglas de Asociación)	A	L	L	A	A	A
			Estimación de relación causal (Test de Ganger)	A	L	L	A	A	H
			Estimación de relación causal (Modelo de Markov)	A	L	L	A	A	A
			Combinación de grado de fiabilidad	A	L	L	A	A	H

A modo de conclusión, en este capítulo se hizo un análisis comparativo de las diferentes estrategias para la gestión de alarmas en IDS de red basados en firmas, según sus resultados en precisión, sensibilidad y reducción de la carga de alarmas. Las técnicas híbridas muestran unos muy buenos resultados en este respecto. Además, se concluye que los algoritmos de correlación basados en similitud mediante reglas simples y las técnicas de priorización son altamente paralelizables y por tanto escalables en ambientes de grandes volúmenes de datos.

En el siguiente capítulo se define la estrategia híbrida de priorización, usando elementos de estrategias analizadas en este capítulo y justificando cada elección desde el punto de vista de los parámetros de desempeño.

CAPÍTULO 4

Descripción e implementación de la estrategia híbrida para la priorización de alarmas

En este capítulo se describe la implementación de cada una de las fases de la estrategia de priorización propuesta. Se ahonda en el flujo de datos entre cada fase, qué recibe y qué muestra a la salida. Adicionalmente, se remarca la importancia de comparar no sólo el nombre del servicio y o aplicación sino también su versión, que es en últimas lo que ayuda a determinar si es vulnerable o no a cierto tipo de ataque y en todo caso saber si la alarma disparada por el IDS es falsa o verdadera.

4.1. Definición de la estrategia

El objetivo de analizar los antecedentes para clasificar las propuestas según sus resultados en términos de los parámetros de rendimiento como exactitud, precisión, sensibilidad, extensibilidad y capacidad de operación en tiempo real se ha conseguido en el capítulo previo. Es de notar que en la mayor parte de los experimentos y como se puede ver en la tabla 7, se evalúa la capacidad de reducción de alarmas mas no tanto la exactitud y precisión de la estrategia. Es importante recordar que, para tener una estrategia de gestión de alarmas con una operación deseable, las tasas de FP y FN deben ser minimizadas además de maximizar la exactitud, las tasas de TP y TN.

Ahora, teniendo en cuenta los resultados obtenidos en los antecedentes, es claro que la estrategia híbrida propuesta en [77] muestra unos muy buenos resultados en términos de reducción de falsas alarmas y para la estrategia de clasificación que usan, logran una precisión y sensibilidad promedio de 97.25% y 99.3% respectivamente. Esto, sumado al hecho de que la mayoría de estrategias propuestas en los últimos años que muestran resultados ligeramente mejores están basadas en las referencias analizadas en la sección anterior, nos da un aval para tomar la decisión de emplear la arquitectura de esta propuesta como la base de la nuestra.
La arquitectura de la estrategia propuesta en [77] se muestra en la figura 4. Para mayor detalle acerca de la implementación de cada módulo el lector puede referirse a la referencia de este trabajo. Aunque los resultados en términos de precisión, sensibilidad y reducción de la carga de alarmas se obtuvieron implementando la propuesta completa, en nuestro caso nos vamos a centrar en las fases que en la figura 4 se etiquetan como **recolección de alertas** y ***clustering* de alertas**, pues consideramos que son las fases centrales.

A partir de este punto se empieza a describir la estrategia propuesta en este trabajo, cuya arquitectura se puede ver en la figura 5. Teniendo en cuenta que las alertas pueden provenir de múltiples nodos IDS, que además pueden ser de diferentes propietarios, es importante disponer de una etapa de normalización donde se tenga una representación uniforme de las alarmas. Esto es justamente lo que se pretende hacer usando el modelo de datos Formato de Intercambio de Mensajes de Detección de Intrusos (IDMEF por sus siglas en inglés) como camino para establecer una representación estándar para las alertas generadas por los IDS. Este trabajo se realizará mediante la incorporación del SIEM Prelude como normalizador de eventos.
A propósito del módulo de verificación, el objetivo fundamental de la verificación de las alertas es discriminar entre los intentos de intrusión fallidos y exitosos.
La metodología a implementar en esta etapa es la de aplicar una medida de distancia entre cada alarma y cada perfil de amenaza con el fin de determinar si la alarma realmente corresponde a un intento de

intrusión exitoso o con probabilidad de éxito. En el caso particular de [75], la medida de distancia aplicada es la *distancia de Hamming*. El perfil de amenaza, también denominado información de contexto de la red, se construye a partir de los reportes arrojados por el escáner de vulnerabilidades.

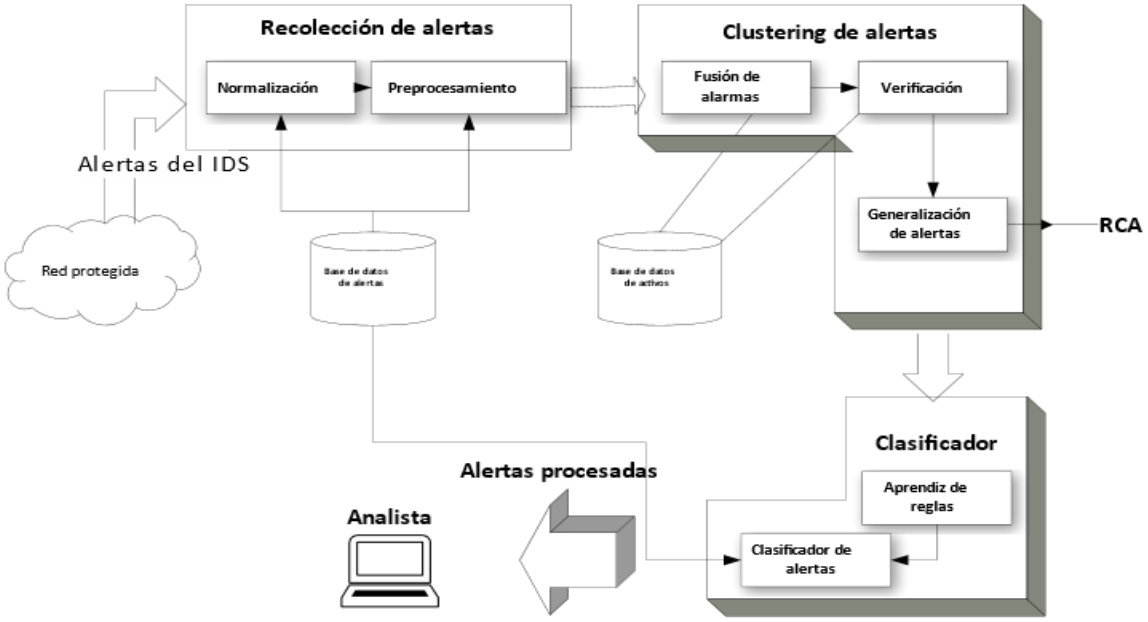

Figura 4. Arquitectura del sistema de *clustering* y clasificación propuesto en [77].

A la salida del módulo de verificación se implementa un módulo de priorización de alarmas. Este módulo de priorización se encarga fundamentalmente de calcular un score que ayude a identificar cuáles son las alarmas sobre las cuales debiera hacerse un análisis prioritario. El score se calcula usando la fórmula de prioridad que usa el Sistema de Gestión de Eventos e Información de Seguridad (SIEM) HP ArcSight [81].

Sintetizando, en la figura 5 se propone un esquema que describe la estrategia propuesta.

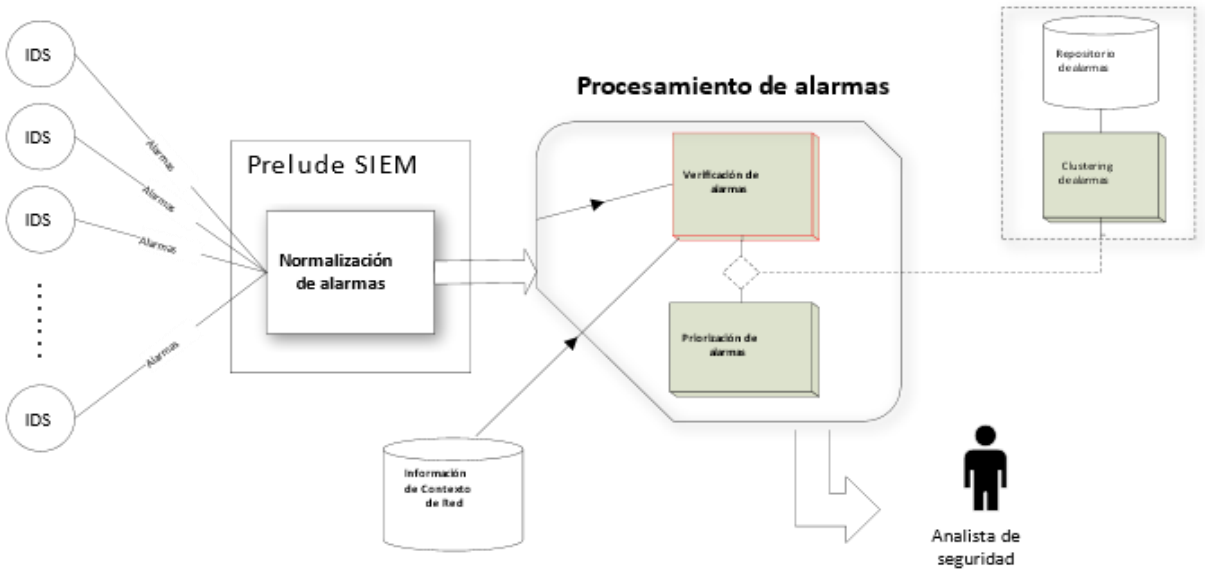

Figura 5. Esquemático de la estrategia de gestión de alarmas propuesta.

Además de la figura anterior, en la figura 6 se muestra un esquemático más descriptivo de la estrategia propuesta. De aquí en adelante se explicará en detalle cada módulo y el flujo entre cada uno.

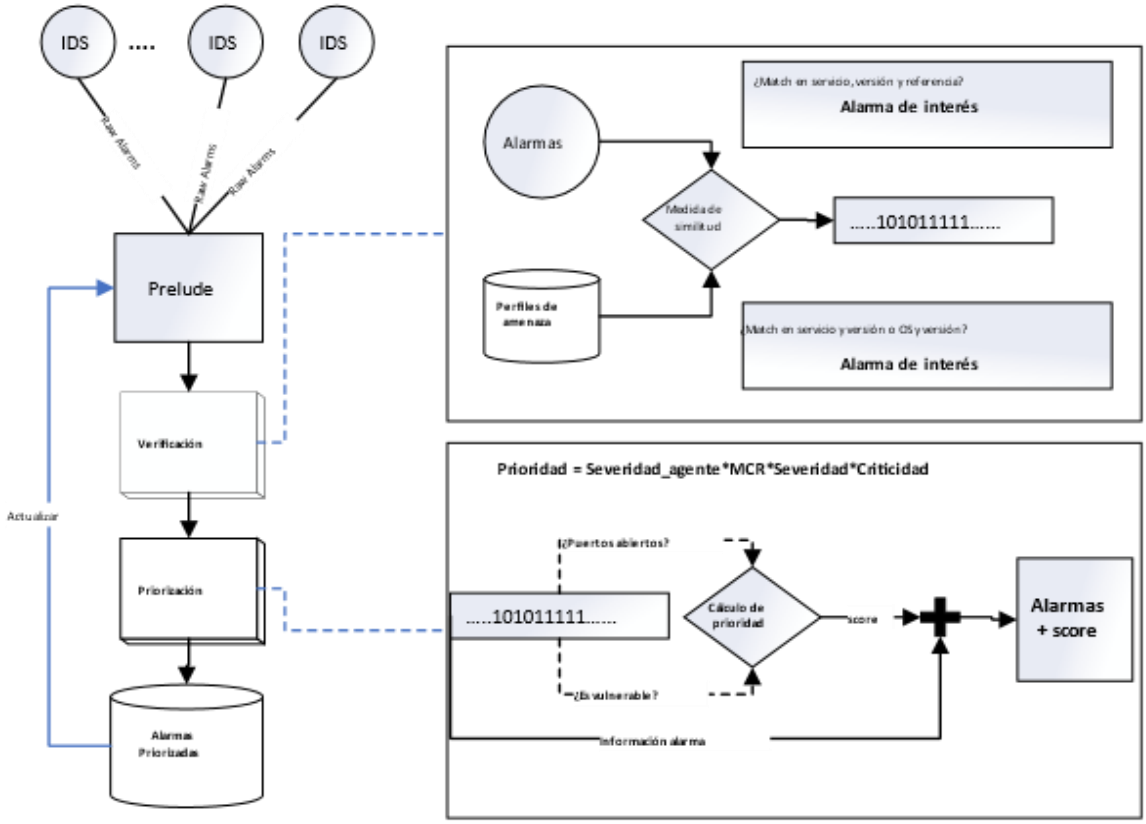

Figura 6. Esquema descriptivo de la estrategia de priorización de alarmas propuesta.

4.1.1. Fase de normalización

Como se mencionó anteriormente, para la normalización de las alarmas se usa Prelude SIEM, que permite la gestión de eventos de seguridad heterogéneos. En la figura 6 se muestra la arquitectura simple de Prelude [82].

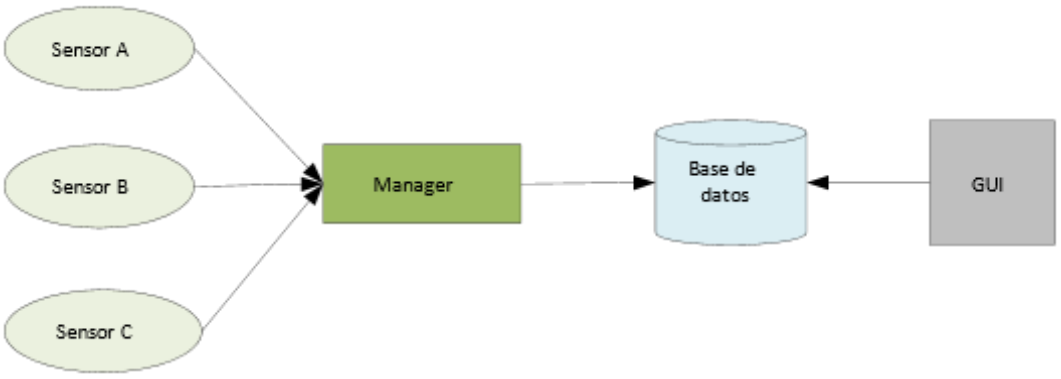

Figura 7. Arquitectura simple de Prelude.

Prelude se divide en varios componentes. Los sensores son responsables de la detección de intrusiones y reportan alertas de manera centralizada utilizando una conexión TLS a un servidor *prelude-manager*. Luego de recibir las alertas, el servidor *prelude-manager* puede procesarlas y enviarlas a un medio especificado por el usuario (base de datos mysql, base de datos postgresql, archivo XML, o cualquier otro formato).
Por defecto, Prelude presenta una compatibilidad nativa con soluciones de seguridad como AuditD, Nepenthes, ufwi-filtered, OSSEC, Pam, Samhain, Sancp, Snort y Suricata. Prelude no depende de una sola marca o formato y es capaz de analizar cualquier tipo de registro (registros del sistema, syslog, archivos sin formato, etc.). Algunas de las plataformas que se pueden beneficiar del motor de análisis de Prelude son: sistemas Unix, conmutadores y enrutadores, firewalls, impresoras etc. [82].

Libprelude es una librería de Prelude que garantiza conexiones seguras entre todos los sensores y el Administrador de Prelude (*Prelude-manager*). *Libprelude* proporciona una interfaz de programación de aplicaciones (API) para la comunicación con los subsistemas de Prelude, proporciona la funcionalidad necesaria para generar y emitir alertas IDMEF.
Esta librería también facilita que el software de terceros sea capaz de comunicarse con los componentes de Prelude. Esta biblioteca proporciona características comunes y útiles utilizadas por cada sensor. Justamente mediante el uso de esta librería es que se habilita la normalización de las alertas generadas por el sensor IDS al formato IDMEF. Estas alarmas normalizadas se almacenan en la base de datos de Prelude. No entraremos en detalle acerca de la estructura de la base de datos de Prelude Manager, pero sí se hará referencia a los atributos de interés que se extrajeron. En la tabla 9 se muestran los atributos de interés para la construcción del perfil de alarmas y su descripción correspondiente.

Tabla 9. Descripción de los atributos usados para la construcción de los perfiles de alarma.

Atributo	Definición
alert_id	Identificador de la alarma asignado por Prelude-Manager
name	Nombre del sensor

type	Tipo del sensor (analizador)
fuente	IP fuente
target	IP destino
time	tiempo de registro de la alarma
sid	ID de la regla asociada con la alarma
port	Puerto destino
srv_name	Nombre del servicio, si se conoce
prot_name	Protocolo de comunicaciones
severity	Severidad asociada con la alarma
i_type	Tipo de impacto
origin	Origen de la referencia de la vulnerabilidad asociada con la alarma
reference	Referencia de la vulnerabilidad

4.1.2. Módulo de verificación

El módulo de verificación recibe dos entradas: las alarmas y la información de contexto de red. La información de contexto de red es una base de datos de perfiles de amenaza construidos a partir de los análisis de vulnerabilidad ejecutados en la red. Cada alarma se correlaciona con los perfiles de amenaza asociados a la dirección IP destino. El resultado de esta correlación es un **vector de correlación** o **vector de verificación**. Este vector de correlación tiene nueve posiciones, donde cada posición refleja el resultado de la comparación de nueve atributos diferentes: tiempo, IP, puerto, protocolo, OS, versión del OS, servicio, versión del servicio y referencia. A continuación, se describe la metodología empleada para la construcción de los vectores de verificación.

En nuestro caso particular, los perfiles de amenaza se construyen a partir del procesamiento de los resultados del análisis de vulnerabilidades arrojado por Nessus Home versión 7.1.3. El escáner activo de Nessus es un software licenciado. En los experimentos se usó una versión libre, con una versión de plugins limitada.

NessusClient 3.2 introdujo un nuevo formato de archivo (.nessus) para escanear, exportar e importar el reporte del VA. Este formato se etiquetó como versión 2 con el lanzamiento de Nessus 4.0. El formato está basado en XML para un fácil procesamiento e implementación [83].
El formato de archivo .nessus enumera dos secciones denominadas *Políticas (Policies) y Reporte (Report)*. Cada sección puede tener múltiples componentes. A continuación, se muestra un esquema básico que incluye el encabezado y pie de página "NessusClientData":

```
<NessusClientData_v2>
<Policy><policyName>My Policy</policyName>
[..]
</Policy>
```

<Report name="My Scan">
[..]
</Report>
</NessusClientData_v2>

Nos vamos a interesar específicamente en la sección de reporte. Esta sección está organizada por un nombre de informe específico e incluye el objetivo y los resultados del análisis.
A continuación se muestra una plantilla de cómo se formatea la sección "Report":

<Report name="Router- Uncredentialed">
<ReportHost name="192.168.0.1">
<HostProperties>
[..]
</HostProperties>
[..]
<ReportItem>
[..]
</ReportItem>
</ReportHost>
</Report>

Dentro de esta sección se encuentran las subsecciones *ReportHost* y *ReportItem*. El componente *ReportHost* dentro de la sección *Report* contiene todos los hallazgos de cada host, incluidos algunos metadatos como la hora de inicio y finalización del análisis, la dirección MAC del hardware asociado, el sistema operativo detectado y un resumen de las vulnerabilidades encontradas por gravedad. Las vulnerabilidades se enumeran por directiva *ReportItem* e incluyen sinopsis de vulnerabilidad, descripción, solución, referencias y salida relevante del complemento.
A partir de los elementos de la sección *Report* se construyen los perfiles de amenaza. Estos perfiles tienen los atributos mostrados en la tabla 10:

Tabla 10. Descripción de los atributos usados para la construcción de los perfiles de amenaza.

Atributo	Definición
Port	Número de puerto.
IP	Dirección IP del host escaneado
Svc_name	Nombre del servicio, si se conoce.
Time	Tiempo de finalización del análisis
OS	Sistema operativo (OS) detectado
Protocol	El protocolo usado para la comunicación (ej. TCP, UDP)
Severity	El nivel de severidad corresponde a un número entre 0 y 4 0 – informativo 1- bajo 2 – medio 3 – alto 4 – crítico
Plugin_id	ID del plugin

Plugin_output	Salida del plugin
Risk_factor	Factor de riesgo asociado con la vulnerabilidad. Los valores están entre bajo, medio y alto.
CVE	Referencia CVE
BID	Referencia a base de datos Bugtraq
CPE	Enumeración de plataforma común
Cvss_base_score	Puntaje base CVSSv2 y/o CVSSv3 (características intrínsecas y fundamentales de una vulnerabilidad que son constantes en el tiempo y en los entornos de usuario)

Una vez definidos los perfiles de alarmas y amenaza, es necesario entrar en consideración acerca de cuáles son los atributos cuya similitud determina la veracidad de la alarma.

En el capítulo 2 se abordaron algunos métodos de correlación entre las alarmas y la información de vulnerabilidades. Como se mencionó allí, se puede identificar la veracidad del evento disparado por el IDS buscando la relación entre el sistema operativo objetivo, la aplicación o servicio y la información de las vulnerabilidades.
Más allá de que se pueda establecer una coincidencia entre el nombre tanto del OS como del servicio, debe hacerse una revisión de las versiones de los mismos, pues es este dato el que me proporciona la certeza de que la vulnerabilidad existe en esa versión del servicio.
La información de la versión del servicio o del OS no se puede extraer directamente a partir de un campo de la alarma. Por ende, es necesario hacer una revisión de la regla que disparó la alarma y cuáles son los sistemas afectados por el ataque asociado a esta regla.
Esta relación de sistemas afectados e id de la firma disparada se estableció de una manera similar a como lo hacen en [84]. El procedimiento consiste en una vez disparada la alarma, extraer el id de firma o signature id y hacer una búsqueda en la base de datos del sensor IDS implementado buscando información adicional acerca de los sistemas afectados.

Considerando el caso particular del IDS Snort, este tiene una base de datos donde se puede consultar la información asociada con cada regla disponible. Esta información se puede consultar en la url *https://www.snort.org/rule-docs/*. Por cada regla consultada se retorna un conjunto de datos divididos en las siguientes secciones:

Message	Mensaje de la alerta
Summary	Resumen de la regla
Impact	Impacto asociado con el factor CVSS
Detailed Information	
Affected systems	Lista de aplicaciones afectadas (CPE y versión)
Easy of attack	Qué tan fácil es ejecutar el ataque
False positives	Reporte de falsos positivos
False negatives	Reporte de falsos negativos
Correction action	Acciones correctivas
Contributors	Contribuyentes
Additional References	Referencias adicionales

El apartado sistemas afectados contiene la información de interés para determinar cuáles son los servicios y/o sistemas operativos afectados, así como la versión de cada uno. Estos datos están

almacenados en descripciones textuales semiestructuradas. El nombre de la aplicación y/o servicio operativo están en formato CPE. CPE es un esquema de nombres estructurado para sistemas, plataformas y paquetes de tecnología de la información. Basado en la sintaxis genérica para los Identificadores Uniformes de Recursos (URI), el CPE incluye un formato de nombre formal, un lenguaje para describir plataformas complejas, un método para comparar nombres con un sistema y un formato de descripción para vincular texto y pruebas a un nombre [85]. CPE usa el siguiente formato:

cpe:/ {part} : {vendor} : {product} : {version} : {update} : {edition} : {language}

Cada campo se define como sigue:

Part – determina el tipo de plataforma usando uno de los siguientes códigos: a = aplicación, h = hardware, o = sistema operativo.
Vendor – Define el nombre del proveedor como "la etiqueta específica más alta de la organización del nombre DNS de la organización" (por ejemplo, apache).
Product – Define el nombre del producto como se especifica en la base de datos CPE, por ejemplo, itunes, quicktime, struts, manageengine etc.

Los siguientes campos son opcionales y completados de acuerdo con cada entrada anterior:

Version – Los números de la versión del producto
Update - El nombre de CPE para la actualización o el paquete de servicio, como "Service Pack 3" en el caso de Windows XP.
Edition - La edición del software, como "pro" para "Professional Edition". Para el hardware, esto también denota la arquitectura, como "i386".
Language: por ejemplo, "inglés" u otro idioma especificado por el software.

Con el propósito de llevar a cabo el proceso de extracción de la información de los sistemas afectados de una forma automatizada se podría seguir lo siguiente: en primer lugar, hay que decir que lo óptimo sería tener la base de datos del IDS Snort a nivel local para no tener que hacer búsquedas a bases de datos en internet y añadir un retardo al proceso. Hay que recordar que, según la definición preliminar de la estrategia, cada que se dispare una alarma se consultaría la base de datos de Snort para extraer la información de los sistemas afectados y añadirla al perfil de la alarma. En la base de datos local, esta información se podría añadir en formato de texto, capturando exactamente la salida de la sección sistemas afectados de la base de datos de Snort, sin hacer ningún procesamiento. Una vez en la base de datos local, esta información se extraería por parte del manager donde se corra la estrategia y se haría el procesamiento correspondiente extrayendo los diferentes tipos de sistemas afectados asociados a cada id de firma.

En el caso particular de la descripción de los sistemas afectados proporcionado por la base de datos de Snort, sólo se cuenta con el producto y la versión, pero no separados por dos puntos como lo muestra el formato CPE, por lo que para extraer nombre y versión por separado habría que pensar en otra alternativa.
Una firma en particular puede tener varios tipos de sistemas afectados. La estrategia para identificar si hay varios tipos de sistemas afectados es sencilla: primero se extrae el nombre del producto haciendo un split cuando se encuentre el primer número. Esto se hace para cada línea leída. La cadena de texto extraída se compara con cada una de las líneas siguientes, en caso de que no sean iguales se asume que hay más de un tipo de sistema afectado. La figura 7 se observa un ejemplo de la sección sistemas afectados.

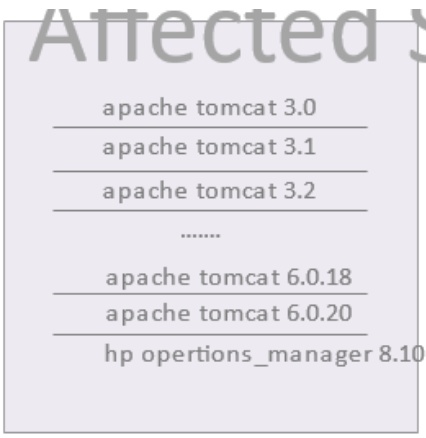

Figura 8. Sistemas afectados asociados con el id de firma de Snort (sid: 1:17156).

Luego de hacer el procesamiento de los datos, la información del OS y la aplicación se almacena en la siguiente estructura de datos:

```
sid_table = {

 'signature_id': [('OS1 / OS2...., OS1_version1, OS1_version2 / OS2_version1...'), ('srv_1 /
                  srv_2 ... ', 'srv_1_version1- srv_1_versionx  / srv_2_version1, srv_2_version2 ... ')],
....
}
```

La estructura de datos mostrada anteriormente no es más que la de un diccionario implementado en Python. La clave del diccionario es el id de firma (signature_id). Por cada clave, se tendrá una lista de dos tuplas. En la primera tupla se almacena la información correspondiente a los sistemas operativos asociados. La tupla tiene dos posiciones. En la primera posición se almacena el nombre del sistema operativo (OS1, OS2, etc.); si hay más de un sistema operativo asociado, estos se separan usando el carácter '/'. En la segunda posición se almacena la información referente a las versiones asociadas al sistema operativo, que también se separan por sistema operativo usando el carácter '/'. En la segunda tupla se almacena la información correspondiente a las aplicaciones asociadas usando el mismo formato para los OS.

Como se mencionó anteriormente, el nombre sea del sistema operativo o de la aplicación está almacenado en formato CPE. Antes de referirnos al formato en el que se almacenan las versiones en el diccionario, es importante precisar qué estructura tiene el número de versión.
Los números de versión y los nombres de versión son usados para proporcionar identificadores únicos de un estado de desarrollo de software y son comunes a prácticamente todo el software. Una forma muy común de presentar los números de versión es a través de un identificador basado en una secuencia como 7.0.8.1. Cada número secuencial tiene significados especiales, que pueden variar completamente dependiendo del autor del software. Teniendo como referencia la secuencia 7.0.8.1, este número de versión se podría interpretar de la siguiente manera [84]:

• El primer número (7), generalmente se refiere a una versión principal del software, a menudo incrementada después de que se hayan realizado cambios importantes en el software.
• El segundo número (0) se refiere a una versión menor dentro de esta versión principal.
• El tercer número (8) generalmente se refiere a un número de compilación específico dentro de la versión menor, y a menudo es más significativo para un desarrollador que para los usuarios finales.

• El cuarto número (1) podría luego referirse a una revisión de esta construcción. La naturaleza de las representaciones numéricas secuenciales facilita la comparación de todas ellas asumiendo que los formatos de fecha están escritos en orden descendente.

En la tabla 11 se muestran los formatos más comunes para representar las versiones de sistemas operativos y/o versiones.

Tabla 11. Formatos comunes para representar las versiones de sistemas operativos y/o aplicaciones [84].

Formato	Descripción
9.0.7.1	Un formato simple
9.0.7b2	La letra b y el número 2 podrían significar que esta es una versión beta con algunos errores corregidos
9.0.7rc1	rc1 usualmente significa candidato de lanzamiento 1
9.0.7r2	r2 significa que este es un lanzamiento comercial con algunas mejoras
9.0_7.1	Otra manera de separar los números de versión
Vista	Nombre usado como un identificador de versión que se refiere a Windows Vista.
Office 2016	Nombre y año usados como identificador de versión
Vista SP2	Nombre usado como identificador de versión, con una segunda porción definiendo una versión menor.
2.5.10/build 69	un número de versión que proporciona un significado claro de los últimos dígitos
2012.05.13	Una fecha usada como un número de versión

Teniendo clara cuál es la estructura de un número de versión, pasamos a describir en qué formato se almacenan en el diccionario.
Un id de firma puede estar asociado con varias versiones de un mismo servicio. Si se tienen más de 4 versiones asociadas, se usa el caracter '-' para resumir la información de versión usando la estructura *versión_menor* '-' *version_mayor.*
Lo primero que se hace es identificar cual es la versión menor y la mayor haciendo comparaciones entre los números de versión. Una vez identificados, se pone la versión menor separada por el identificador '-' que en este contexto traduciría "hasta", y luego se pone la versión final. Para ilustrar lo anterior, supongamos que el sistema afectado es *apache tomcat* con versiones 3.0, 3.1, 3.1.1, 3.2, 3.2.1, 3.2.2, …., 6.0.20. Usando la lógica anteriormente descrita, el servicio quedaría expresado como *apache tomcat 3.0 – 6.0.20.*

Si la cantidad de versiones son menores o iguales a cuatro, simplemente se separan por comas. Una versión en particular puede tener el caracter '*'. Supongamos el número de versión 4.5.* . En este caso, la lectura que se hace a partir de la inclusión del asterisco, '*', es que se incluyen todas las versiones con los dos primeros identificadores iguales a 4 y 5 respectivamente, por lo que la comparación sólo se establecería con los dos primeros identificadores de versión, sin importar el último.
La información del diccionario será añadida a cada perfil de alarma según el identificador de firma.

4.2. Generación de vectores de correlación

Para la generación de los vectores de correlación, lo primero que se hace es extraer los atributos de interés de cada perfil, tanto de alarma como de amenaza, ya que no todos son compartidos por ambos perfiles. La razón por la cual no se han generado los perfiles considerando únicamente los atributos compartidos por ambos, es debido a que los demás atributos añadidos podrían constituir parte de un análisis adicional que potencialmente podría añadírsele a la estrategia.
La comparación de los atributos compartidos entre los perfiles se establece como muestra la tabla 12.

Tabla 12. Comparación entre el perfil de alarma y el perfil de amenaza indicando en rojo las posiciones del vector con más peso.

Alarma		**Perfil de amenaza**	
Time	>=	Time	1 sino 0
IP	==	IP	1 sino 0
Protocol	==	Protocol	1 sino 0
Port	==	Port	1 sino 0
OS	==	OS	1 sino 0
OS_version	==	OS_version	1 sino 0
Service	==	Service	1 sino 0
Service_version	==	Service_version	1 sino 0
reference	==	reference	1 sino 0

La razón por la cual se acepta un tiempo de alarma mayor al tiempo del perfil de amenaza es porque si se encuentra que la alarma es de interés habrá certeza de que se los perfiles están actualizados.
Es evidente que el caso ideal es cuando resulta un vector de solo 1's. Si es así, habrá certeza de que la alarma es un indicador real de ataque. Sin embargo, hay otros casos que son de interés y que consideraremos a continuación:

Caso 1: vector resultante 11XX1X111

En este caso habrá coincidencia en tiempo, IP, tipo de sistema operativo, servicio y la versión del servicio, por lo que la alarma se clasificará como alarma de alto interés.

Caso 2: vector resultante 111100000

Si hay coincidencia en los primeros cuatro atributos la alarma se clasificará como factible, pero no de interés por lo cual requerirá, si así se considerase, de una revisión adicional.

Caso 3: cualquier vector en el que la segunda posición sea cero

En este caso no habrá coincidencia en la IP por lo que la alarma será descartada.

Caso 4: cualquier vector en el que la primera posición sea cero

En este caso la alarma se clasificará según su coincidencia en los demás atributos, pero se elevará un mensaje solicitando la actualización de los perfiles de amenaza.

Para cada perfil de alarma se calculará el vector de correlación con los perfiles de amenaza asociados con la dirección IP objetivo de la alarma. Lo anterior establece el primer filtro para las alarmas y garantiza que el producto de la comparación de la dirección IP siempre será 1. Esta característica es deseable puesto que se está reduciendo el conjunto de perfiles de amenaza y por tanto el tiempo de comparación y generación del vector de correlación. Incluso podrían idearse otros métodos de filtrado para mejorar el tiempo de cómputo de un vector de correlación.

Dado que cada alarma generará un vector de correlación por cada perfil de amenaza en el conjunto de interés, habrá que establecer un mecanismo para seleccionar un solo vector de correlación que registre la máxima coincidencia. El mecanismo de selección es como sigue: todos los vectores cuyas últimas cinco posiciones sean 1 se almacena en una lista temporal. Se retornará aquel en el que haya mayor número de coincidencias en el resto de posiciones. Si todas las posiciones no son 1, se buscará una combinación en la que las posiciones correspondientes al sistema operativo y su versión o al servicio y su versión sean 1; a partir de aquí se usará el mismo procedimiento que en el primer caso. Haciendo lo anterior estamos dando prioridad a aquellas alarmas que tengan coincidencia en sistema operativo o aplicación.

4.3. Cálculo de prioridad

La entrada al módulo de prioridad es un conjunto de vectores producto de la verificación de las alarmas con los perfiles de amenaza. A cada vector se le añade el identificador único de alarma añadido por Prelude y el id de firma. Luego, se le asigna un score de prioridad usando la fórmula de prioridad del SIEM HP Arcsight. ArcSight ESM emplea un proceso de priorización de nivel de amenaza para cada evento con el fin de reducir los falsos positivos y enfocar a los analistas en la amenaza de mayor prioridad. La fórmula es el producto de cuatro factores de prioridad descritos a continuación.

4.3.1. Factores de prioridad

La fórmula de prioridad consta de cuatro factores que se utilizan para calcular un score de prioridad. Todos los valores de los factores de prioridad están dentro de un rango de 0 a 10, donde 0 es bajo y 10 es alto. Un factor de alta prioridad generalmente indica un evento con un factor de riesgo más alto. En la tabla 13 se describen los factores de prioridad, los casos posibles y los puntajes asignados según el caso [86].

Tabla 13. Definición de los factores de prioridad de la fórmula de HP Arsight [86].

Confianza del modelo (Model Confidence)	**La confianza del modelo se refiere a si el activo amenazado se ha modelado o no en ArcSight. Puntuación máxima = 10.**
+4	El activo objetivo se modela en ArcSight y su ID de activo está presente.
+4	El activo objetivo ha sido escaneado en busca de puertos abiertos
+4	El activo objetivo ha sido escaneado en busca de vulnerabilidades

Relevancia (Relevance)	La relevancia se refiere a si un evento es o no relevante para un activo en función de si el evento contiene puertos y/o vulnerabilidades conocidas, y si es así, si esas vulnerabilidades y/o puertos están expuestos en el activo. Si un activo no expone las vulnerabilidades o puertos contenidos en el evento, el evento no es relevante para el activo. El valor por defecto es de 10 (puntuación máxima).
-5	Puertos: evento no contiene puerto.
-5	Puertos: activo de destino no analizado para puertos abiertos.
+5	Puertos: el puerto escaneado está abierto en el activo de destino.
-5	Vulnerabilidad: activo objetivo no analizado en busca de vulnerabilidades.
+5	Vulnerabilidad: vulnerabilidad expuesta del activo objetivo en el escaneo.
Severidad (Severity)	La severidad puede ser vista como una función de historia. ¿Se ha atacado el sistema anteriormente, se ha comprometido antes, o el atacante ha escaneado o atacado la red en el pasado? Las diferentes puntuaciones se asignan en función de la presencia del atacante y del objetivo en una de las listas de sistemas activos de ArcSight (/ Todas las listas activas / Listas activas del sistema / ...), cuyo contenido generalmente se actualiza automáticamente mediante las reglas de ArcSight. Puntuación máxima = 10.
+6	El activo aparece como un atacante en la lista de infiltrados.
+5	El activo aparece como un atacante en la lista hostil.
+3	El activo aparece como un objetivo en la lista de comprometidos.
+3	El activo aparece como un atacante en la lista de sospechosos.
+1	El activo aparece como un atacante en la lista de reconocimiento.
Criticidad de activos (Asset Criticality)	La criticidad de los activos mide la importancia que tiene el activo objetivo según lo establecido en el proceso de modelado de la red mediante el uso de las categorías de activos de inventario. Por ejemplo, los sistemas o dispositivos orientados al cliente con acceso a información confidencial se clasificarían como nivel de criticidad Alto, mientras que un sistema de prueba o de clasificación puede tener un nivel de criticidad Bajo. Puntuación máxima = 10.
+10	El activo se encuentra clasificado con criticidad Muy alta
+8	El activo se encuentra clasificado con criticidad Alto
+6	El activo se encuentra clasificado con criticidad Medio
+4	El activo se encuentra clasificado con criticidad Bajo
+2	El activo se encuentra clasificado con criticidad Muy bajo
+0	El activo no entra en ninguna de las categorías anteriores.

Los valores asignados a los factores de prioridad se utilizan para determinar la prioridad del evento generado (en nuestro caso las alarmas del IDS) según los cálculos que se describen a continuación.

Tabla 14. Cálculo de prioridad usando la fórmula de HP Arsight [86].

Factor de nivel de amenaza	Factor de ponderación
Severidad del evento (AgentSeverity)	Los conectores informan los valores de gravedad según el dispositivo, la situación y su configuración. Los valores van de 0 a 10, y se muestran como Desconocido (2), Bajo (4), Medio (6), Alto (8) y Muy alto (10). [En nuestro caso estos valores presentaran una ligera variación. Esto se muestra más adelante]

<table>
<tr><td>Confianza del modelo y relevancia</td><td>La confianza y relevancia del modelo (MCR) tienen un rango de 0 a 10 y se combinan para dar un valor neto en el rango de 0 a 10. Por ejemplo, un Modelo de 10 y una Relevancia de 5 devolverían un valor de 0.5. Este factor combinado establece el grado de compatibilidad con el valor de severidad original del evento. La fórmula real para calcular el factor combinado MCR es:
$$MCR = \frac{R}{(R+M) - (\frac{R*M}{10})}$$</td></tr>
<tr><td>Severidad (Severity)</td><td>Este factor refleja la severidad del ataque potencial que representa un evento. La severidad tiene un rango de 0 a 10, con el valor más alto (10) agregando un factor de peso del 30%. La fórmula utilizada para calcular el valor de la severidad (S) es:
$$\frac{(1 + S * 3)}{100}$$</td></tr>
<tr><td>Criticidad</td><td>La criticidad de activos varía de 0 a 10, y puede aumentar o disminuir un valor de prioridad hasta en un 20%. Un valor de 8 es el punto medio, con un efecto de cero (0) por ciento, mientras que un 0 resta el 20%. Esto se refleja en el siguiente cuadro
<table>
<tr><th>Valor</th><th>Factor de peso</th></tr>
<tr><td>10</td><td>+20%</td></tr>
<tr><td>8</td><td>No hay cambio</td></tr>
<tr><td>6</td><td>-5%</td></tr>
<tr><td>4</td><td>-10%</td></tr>
<tr><td>2</td><td>-15%</td></tr>
<tr><td>0</td><td>-20%</td></tr>
</table>
$$1 + (C - \frac{8}{10}) * 20\%$$</td></tr>
<tr><td>Cálculo de prioridad</td><td>$$\boldsymbol{Prioridad = severidad\_del\_agente * MCR * Severidad * Criticidad}$$</td></tr>
</table>

A la severidad del agente o *agentSeverity* se le asigna un valor según la severidad reportada por la alarma que dispara Snort. La tabla 15 muestra los posibles valores [87]:

Tabla 15. Valores de severidad asignados por Snort según la clase de ataque detectado.

Clase	Descripción	Prioridad
attempted-admin	Intento para ganar privilegios de administrador	Alta
attempted-user	Intento de ganar los privilegios de usuario	Alta

inappropriate-content	Se detecto contenido inapropiado	Alta
policy-violation	Potencial violación a la privacidad corporativa	Alta
shellcode-detect	Se detectó codigo ejecutable	Alta
successful-admin	Ganancia de privilegios de administrador exitosa	Alta
successful-user	Ganancia de privilegios de usuario exitosa	Alta
trojan-activity	Fue detectado un troyano	Alta
unsuccessful-user	Ganancia de privilegios de usario no exitosa	Alta
web-application-attack	Ataque de aplicación web	Alta
attempted-dos	Intento de denegacion de servicio	Media
attempted-recon	Intento de fuga de información	Media
bad-unknown	Tráfico potencialmente malo	Media
default-login-attempt	Intento de iniciar sesión con un nombre de usuario y contraseña predeterminados	Media
denial-of-service	Detección de un ataque de denegación de servicio	Media
misc-attack	Ataque Misc	Media
non-standard-protocol	Detección de un protocolo o evento no estándar	Media
rpc-portmap-decode	Decodificación de una consulta RPC	Media
successful-dos	Denegación de servicios	Media
successful-recon-largescale	Fuga de información a gran escala	Media
successful-recon-limited	Fuga de información	Media
suspicious-filename-detect	Se detectó un nombre de archivo sospechoso	Media
suspicious-login	Se detectó un intento de inicio de sesión utilizando un nombre de usuario sospechoso	Media
system-call-detect	Se detectó una llamada al sistema	Media
unusual-client-port-connection	Un cliente estaba usando un puerto inusual	Media
web-application-activity	Acceso a una aplicación web potencialmente vulnerable	Media
icmp-event	Evento ICMP genérico	baja
misc-activity	Actividad misc	baja
network-scan	Detección de un escaneo de red	baja
not-suspicious	Tráfico no sospechoso	baja
protocol-command-decode	Protocolo genérico de decodificación de comandos	baja

string-detect	Se detectó un string sospechoso	baja
unknown	Tráfico desconocido	baja
tcp-connection	Se detectó una conexión TCP	Muy baja

Donde el valor numérico se establece así:

Tabla 16. Mapeo del valor de severidad asignado por Snort.

Severidad	**Equivalente numérico**
Alta	4
Media	3
Baja	2
Muy baja	1

Según lo anterior, se puede notar que los valores que toma el *agentSeverity* son de 1 a 4 y no de 0 a 10 como se muestra en la tabla 14. Esto es una ligera modificación que se le hace a la fórmula adaptándola al ambiente propuesto que no modifica la lectura que se hace del score de prioridad.

Entendiendo que en la definición de la estrategia no se usa HP Arsight, para el valor de confianza del modelo (MC) se define lo siguiente:
Si se encuentra que la posición 3 del vector de correlación es 1, es decir que se encontró coincidencia en el puerto, se puede leer que el puerto en cuestión está abierto en el activo objetivo. Además, si la posición 9 del vector de correlación es 1, es decir si se ha encontrado una coincidencia en la referencia a una vulnerabilidad, o si las posiciones 7 y 8 o 5 y 6 son 1 (si se encuentra una coincidencia en OS y versión o servicio y versión) se asume que el servicio es vulnerable por lo que se asigna un valor de 10 al factor MC y un valor de 10 al factor R. Estos valores no se asignan de manera arbitraria. Para su asignación se sigue lo que sugiere en la definición de la fórmula [81,86]. Cuando el activo se modela en Arcsight, el puerto asociado con el evento está abierto y se encuentra vulnerable, al factor MC se le asigna un valor de 10. Asimismo, si el puerto asociado con el evento se encuentra abierto (arriba), y el servicio asociado con este puerto se encuentra vulnerable, entonces al factor R se le asigna un valor de 10.
Si ninguno de los dos casos anteriores se cumple, pero se detecta el puerto abierto entonces se asigna un valor de 8 al factor MC y un valor de 5 al factor R.
Por último, si el puerto se encuentra cerrado entonces al factor MC se le asigna el valor de 4 y al factor R el valor de cero. Lo anterior se sintetiza en la figura 8.

El plan original en el diseño de la estrategia era simplemente asignar un valor a cada alerta, indicando si era probable que fuera un falso positivo o un positivo verdadero y en todo caso que descartara los falsos positivos. Sin embargo, a medida que se desarrollaba el sistema, se reconocía el hecho de que las alertas no siempre podían agruparse fácilmente en dos categorías. Este hecho motivó a que, en lugar de implementar un clasificador binario, se implementara un clasificador o asignador de prioridades multi clase. En la figura 8 se muestra el esquemático descriptivo de la propuesta completa. En el próximo capítulo se registran los resultados obtenidos al implementar el módulo de priorización.

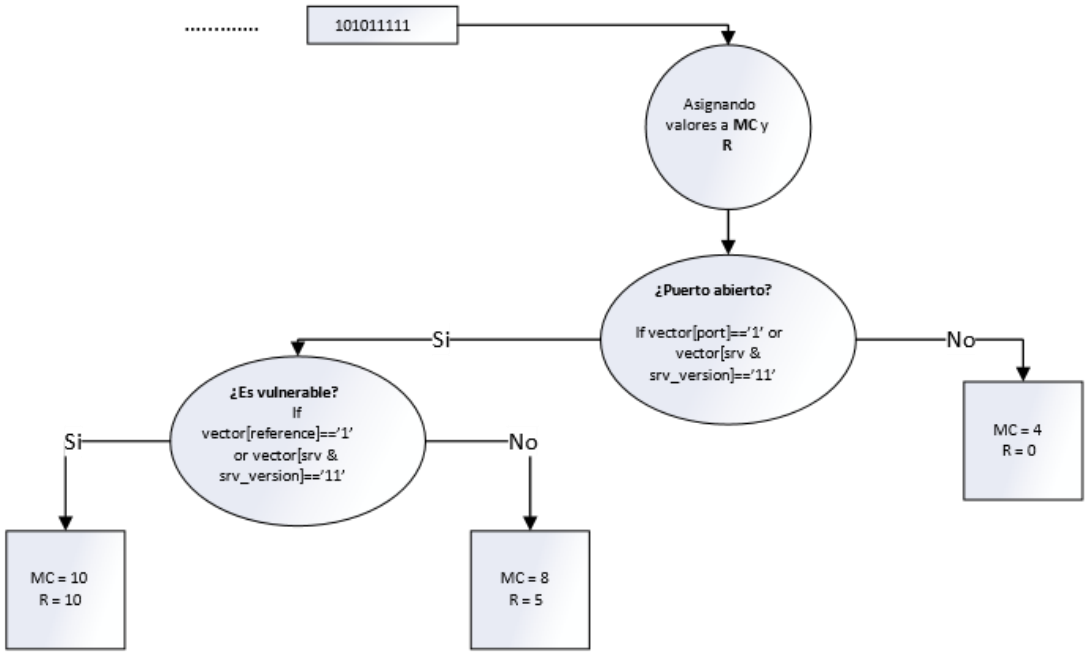

Figura 9. Lógica para asignación de valores a factores de prioridad MC y R. Donde MC se refiere a confianza del modelo y R significa relevancia.

CAPÍTULO 5

Escenario de pruebas

En este capítulo se describe cada una de las herramientas usadas en el escenario de pruebas para evaluar la estrategia de priorización.

5.1. Topología de red y descripción de dispositivos

Para evaluar la capacidad de la estrategia propuesta para priorizar las alertas relacionadas con ataques reales, se diseñó el experimento con la arquitectura de red mostrada en la figura 10.

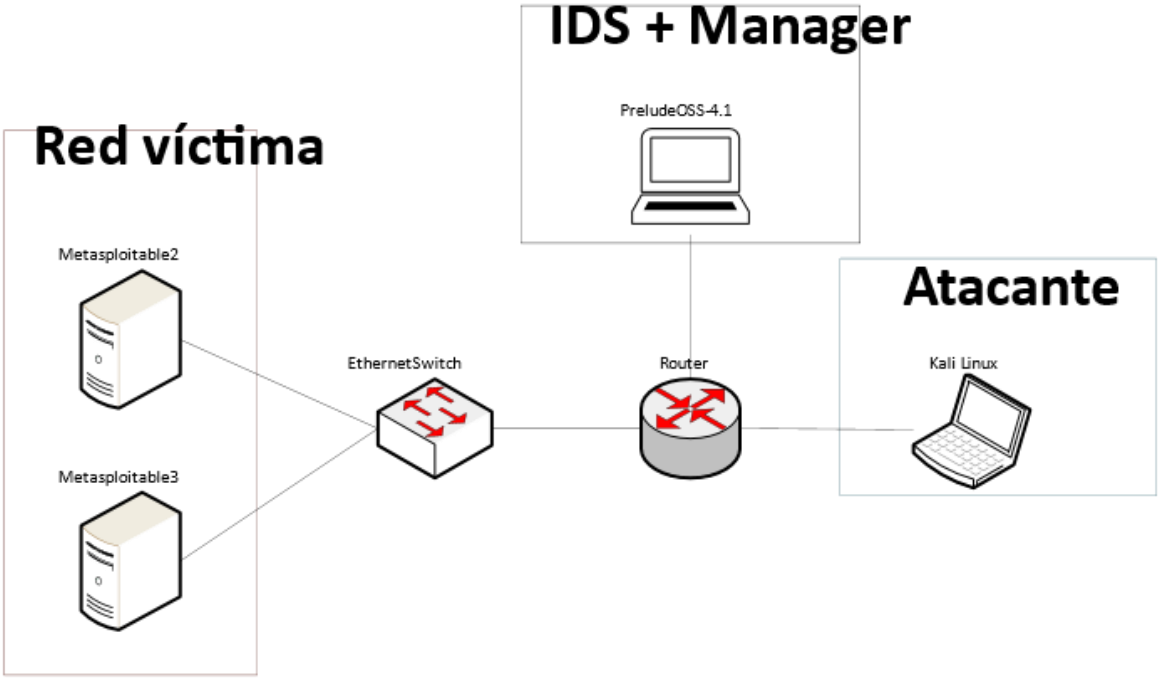

Figura 10. Arquitectura de red propuesta para la fase experimental.

El IDS y el manager de Prelude corren en la misma máquina. Todo el tráfico de ataque se mapea al puerto span al que está conectada la máquina donde corren el proceso de Snort y Prelude. El hecho de usar dos servidores es arbitrario y supone un escenario donde se tienen dos objetivos, no sólo uno. Adicionalmente, se usan Metasploitable y Metasploitable 3 puesto que son máquinas con información documentada acerca de sus vulnerabilidades, lo que permite hacer una comparación de las vulnerabilidades instaladas con las detectadas por el escáner de Nessus.
Este tipo de escenario es adecuado para la evaluación de la estrategia puesto que muestra una exposición directa de la red víctima con el atacante (están dentro de la misma subred) y permite analizar y ver en el IDS todo el tráfico de ataque. Esta no es una situación real, puesto que en la realidad los servidores en la red operativa no deberían contar con tantas vulnerabilidades. Además, el

acceso a estos servidores se protege usando equipos de seguridad perimetrales como Firewalls además de DMZs. Dicho lo anterior, es momento de describir los equipos usados en el experimento.

La topología de red se simuló usando el software GNS3 [88] versión 2.1.11. Para la virtualización de los OS se usó Oracle VM VirtualBox 5.2.20 r125813 (Qt5.6.1). En la tabla 17 se describen los dispositivos de red usados:

Tabla 17. Descripción de los dispositivos usados en el escenario de prueba.

Dispositivo (Versión de OS y kernel)	**Función dentro de la arquitectura**	**IP**
Kali Linux versión de Kernel 4.18.0-kali1-amd64	Atacante	192.168.1.7
Prelude OSS 4.1 sistema operativo CentOS Linux 7 con kernel .10.0-862.11.6.e17.x86_64	SIEM y sistema donde se corre Snort	192.168.1.5
Metasploitable2 con sistema operativo Ubuntu 8.04 (gutsy) y kernel Linux 2.6	Servidor víctima	192.168.1.19
Metasploitable3 con sistema operativo Microsoft Windows Server 2008 R2 Standard Service Pack 1	Servidor víctima	192.168.1.21
Cisco c3600 con un módulo de red NM-16ESW	Switch con port mirroring para duplicar los paquetes vistos en la interfaz conectada al atacante y analizarlos con Snort	Descrito como router en la figura 10.
EthernetSwitch	Switch genérico para conectar los servidores	

Por defecto la máquina virtual de Prelude usada viene con 5 agentes conectados al manager:

Tabla 18. Agentes conectados por defecto al manager de Prelude

Agente	**Función**
Prelude LML	Analizador de logs
Prelude Manager	Concentrador
Ossec	IDS de host, chequeador de integridad de archivos y analizador de logs
Prelude Correlator	Correlador
Suricata	NIDS

Para registrar el agente Snort con el manager de Prelude se siguieron los pasos sugeridos en la documentación de Prelude [89]. La versión de Snort que se instaló fue la 2.9.11.1. Con Snort se generan los archivos pcap que luego son procesados nuevamente por Snort para generar los archivos unified2. Estos últimos son procesados por Barnyard2 que tiene el plugin de salida a Prelude.

Mediante Barnyard se cargan las alarmas en la base de datos de Prelude. Ya en la base de datos de Prelude, las alarmas se pueden chequear usando la interfaz web Prewikka. En la figura 11 se puede ver una captura de pantalla de las alarmas asociadas con el agente Snort mostradas en Prewikka.

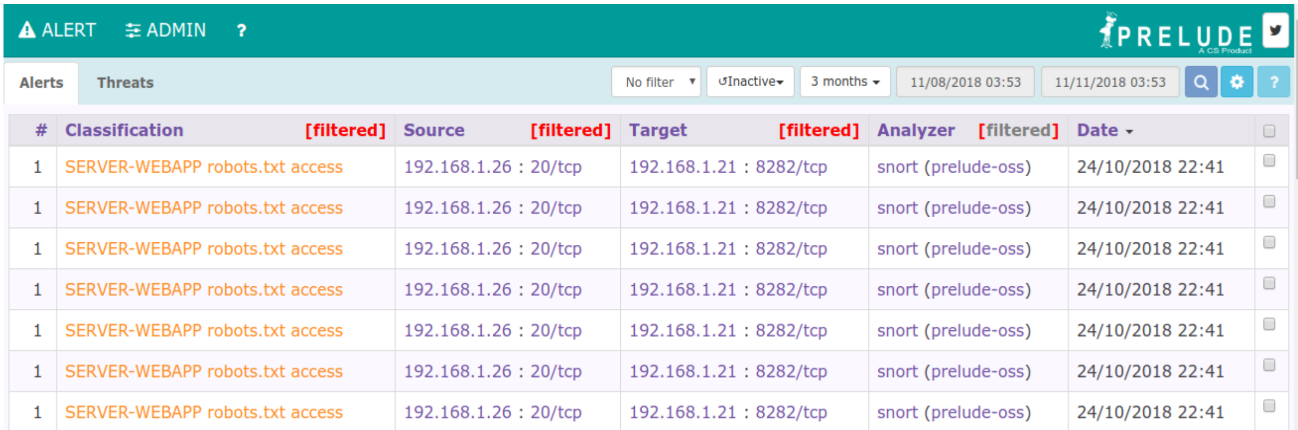

Figura 11. Interfaz web Prewikka.

Para explotar las vulnerabilidades presentes en los dos servidores se usa el framework Metasploit, instalado en la versión de Kali Linux usada.

Metasploit Framework

Metasploit Framework [90] es una herramienta muy potente, basada en código libre, que permite hacer tests de penetración en máquinas remotas. Cuenta con una gran base de datos de exploits con la que se pueden detectar y explotar vulnerabilidades de seguridad en el sistema objetivo.

5.2. Mestasploitable2 y vulnerabilidades asociadas

Metasploitable es un servidor Ubuntu 8.04 que se instala en una imagen de VMWare 6.5 con varios paquetes vulnerables incluidos. Existen guías variadas que describen paso a paso el proceso de explotación de las aplicaciones vulnerables en este servidor. Para la explotación de las aplicaciones vulnerables de interés, se siguieron los pasos descritos en las referencias citadas [91-94].

Los ataques realizados se concentraron en explotar las vulnerabilidades asociadas en la tabla 19. Para cada vulnerabilidad se buscó la regla de Snort asociada con el ataque que la explota. Esta búsqueda se hizo usando la base de datos de ***Security-database*** [95]**.** En esta base de datos también se relacionan los plugin de Nessus para detectar este tipo de vulnerabilidades. En la mayoría de casos hay más de un plugin asociado para detectar cada tipo de vulnerabilidad, en otros, no se encuentran un plugin que detecte la vulnerabilidad directamente. El CPE se extrae de la base de datos de Snort, buscando por id de regla.

5.3. Metasploitable3 y vulnerabilidades asociadas

Metasploitable3 es una máquina virtual construida desde cero con una gran cantidad de vulnerabilidades de seguridad. Está destinado a ser utilizado como un objetivo para probar explotaciones con metasploit. En [96] se listan algunas de las vulnerabilidades añadidas y los módulos de metasploit para explotarlas. Para la explotación se siguieron los pasos descritos en las

referencias [97-100]. En la tabla 16 se listan las vulnerabilidades explotadas en este experimento, así como las referencias a las bases de datos de CVE, el plugin de Nessus asociado y el id de firma de Snort.

Tabla 19. Relación entre las vulnerabilidades de metasploitable2 asociadas con las referencias a bases de datos de vulnerabilidades, regla de Snort y plugin id de Nessus.

Ataque o Servicio/Aplicación afectado	**CPE + versión (Sistemas afectados)**	**Referencias**	**Id de firma (Snort)**	**Nessus_plugin id**
Java RMI registry	sun jdk 1.5.0, 1.6.0 / sun jre 1.5.0, 1.6.0	CVE-2011-3556, CVE-2010-0094	21387	45474 – 46295 - 60776
Vsftpd smiley face backdoor		OSVDB: 73573BID: 48539	19415	
PHP Argument Injection CGI	php 1.0 - 5.4.2	CVE-2012-1823	22063	70728
UnrealIRCD 3.2.8.1 Backdoor Command Execution	unrealircd unrealircd 3.2.8.1	CVE-2010-2075	25106	46882
Samba Symlink Directory Traversal		CVE-2010-0926	44487, 44485, 44489	44406

Tabla 20. Relación entre las vulnerabilidades de metasploitable3 asociadas con las referencias a bases de datos de vulnerabilidades, regla de Snort y plugin id de Nessus.

Servicio/Aplicación	**CPE**	**Referencias**	**Id de firma (Snort)**	**Nessus_plugin id**
Apache Struts 2.3.20, 2.3.20.1, 2.3.24, 2.3.24.1, 2.3.28	apache struts 2.3.20, 2.3.20.1,2.3.24, 2.3.24.1, 2.3.28	CVE-2016-3087	39190	90773
Apache Tomcat 3.0.x – 6.0.20	apache tomcat 3.0.* - 6.0.20 / hp operations_manager 8.10	CVE-2009-3843 CVE-2009-4189	17156	
manageengine desktop_central 9.0	Manageengine desktop_central:9.0	CVE-2015-8249	34718	

Elasticsearch 1.1.1	Elasticsearch 1.1.1	CVE-2014-3120	36256	76572
Apache Axis2 1.3 - 1.6	apache axis2 1.3 - 1.6	CVE-2010-0219 BID: 12141, OSVDB:397	18985	46740
rubyonrails:web console 2.1.2		CVE-2015-3224	40332	84452, 84255
Java_jmx_server	vmware:vcenter_server:5.0-6.0	CVE-2015-2342	36532	86255

Además de usar metasploit para explotar las vulnerabilidades, se usaron Ostinato versión 0.9 Revision y Scapy versión 2.4.0 para generar tráfico de background que disparara alarmas no relevantes o en su defecto falsas alarmas. Para lograr lo anterior, a cada paquete se le configuraba la fuente y el destino, el protocolo capa 2, capa 3 (IP), capa 4 (TCP), puerto destino de interés y en la carga útil (payload) se adicionaba un patrón de texto o información relacionado con el patrón de detección de alguna de las reglas en la base de datos de Snort. Con esto, lográbamos disparar alarmas sin ejecutar ataques reales.

La secuencia de ataques fue aleatoria. Es decir, en un momento dado se atacaba el servidor Ubuntu y luego se generaba tráfico de background para disparar falsas alarmas. En otro momento se atacaba el servidor Windows y luego el Ubuntu para luego generar tráfico de background. En general, la cantidad de paquetes generados con Scapy y Ostinato oscilaba entre 10 y 100. De ahí que la cantidad de alarmas generadas para un tráfico particular fuera equivalente a 10 o 100. Como es bien sabido, dependiendo del servicio que se quiera atacar el tipo de tráfico tendrá unas características específicas como protocolo, puerto, cabeceras y payload propias de las capas superiores en el modelo TCP/IP. Con lo anterior, mediante la herramienta metasploit se generaba el tipo de tráfico con el que se levantaban las alarmas asociadas con ataques reales.

La lista de alarmas generadas se muestra en la tabla 21.

Tabla 21. Lista de alarmas disparadas por Snort.

Id de firma	**Firma**	**Severidad Snort**	**IP del objetivo**	**# de alarmas**
1:1852	SERVER-WEBAPP robots.txt access	media	192.168.1.21	100
1:18096	SERVER-APACHE Apache Tomcat username enumeration attempt	media	192.168.1.21	100
1:32672	SERVER-OTHER Cisco ios ftp proxy overflow attempt	alta	192.168.1.21	100
1:2307	SERVER-WEBAPP PayPal Storefront remote file include attempt	alta	192.168.1.21	100
129:2	stream5: Data on SYN packet	baja	192.168.1.21	100
1:2307	SERVER-WEBAPP PayPal Storefront remote file include attempt	alta	192.168.1.16	10

1:977	SERVER-IIS .cnf access	media	192.168.1.21	10
1:17502	SERVER-APACHE Apache Tomcat UNIX platform directory traversal	alta	192.168.1.21	10
1:12711	SERVER-APACHE Apache Tomcat WebDAV system tag remote file disclosure attempt	Media	192.168.1.21	10
1:29456	PROTOCOL-ICMP Unusual PING detected	media	192.168.1.21	11
1:366	PROTOCOL-ICMP PING Unix	baja	192.168.1.21	11
1:384	PROTOCOL-ICMP PING	baja	192.168.1.21	11
1:40063	OS-LINUX Linux Kernel Challenge ACK provocation attempt	alta	192.168.1.28	400
1:18096	SERVER-APACHE Apache Tomcat username enumeration attempt	media	192.168.1.16	100
119:33	http_inspect: UNESCAPED SPACE IN HTTP URI	baja	192.168.1.26	1
123:8	frag3: Fragmentation overlap	baja	192.168.1.21	17
123:3	frag3: Short fragment, possible DoS attempt	baja	192.168.1.21	336
1:384	PROTOCOL-ICMP PING	baja	192.168.1.28	5
123:8	frag3: Fragmentation overlap	baja	192.168.1.21	267
1:1882	INDICATOR-COMPROMISE id check returned userid	media	192.168.1.26	2
1:498	INDICATOR-COMPROMISE id check returned root	media	192.168.1.26	2
1:408	PROTOCOL-ICMP Echo Reply	baja	192.168.1.26	6
1:29456	PROTOCOL-ICMP Unusual PING detected	media	192.168.1.28	5
120:3	http_inspect: NO CONTENT-LENGTH OR TRANSFER-ENCODING IN HTTP RESPONSE	baja	192.168.1.21	14
129:15	stream5: Reset outside window	media	192.168.1.21	2
129:12	stream5: TCP Small Segment Threshold Exceeded	media	192.168.1.21	17
129:15	stream5: Reset outside window	media	192.168.1.26	9
1:1882	INDICATOR-COMPROMISE id check returned userid	media	192.168.1.7	18
1:498	INDICATOR-COMPROMISE id check returned root	media	192.168.1.7	18
1:30375	INDICATOR-SHELLCODE Metasploit payload cmd_unix_reverse	alta	192.168.1.19	15
1:10000002	Snort Alert [1:10000002:1]	alta	192.168.1.19	8
1:384	PROTOCOL-ICMP PING	baja	192.168.1.19	14
128:4	Ssh: Protocol mismatch	media	192.168.1.19	1
1:29456	PROTOCOL-ICMP Unusual PING detected	media	192.168.1.19	14
1:366	PROTOCOL-ICMP PING Unix	baja	192.168.1.19	14
1:30383	INDICATOR-SHELLCODE Metasploit payload cmd_unix_reverse_ruby	alta	192.168.1.19	3

1:408	PROTOCOL-ICMP Echo Reply	baja	192.168.1.7	24
1:39190	SERVER-APACHE Apache Struts remote code execution attempt	alta	192.168.1.21	2
1:17156	SERVER-APACHE HP Performance Manager Apache Tomcat policy bypass attempt	alta	192.168.1.21	4
120:3	http_inspect: NO CONTENT-LENGTH OR TRANSFER-ENCODING IN HTTP RESPONSE	baja	192.168.1.7	40
1:32639	EXPLOIT-KIT Sweet Orange exploit kit jar file requested on defined port	alta	192.168.1.7	6
129:15	stream5: Reset outside window	media	192.168.1.19	9
1:31767	SERVER-OTHER MRLG fastping echo reply memory corruption attempt	media	192.168.1.19	4
129:15	stream5: Reset outside window	media	192.168.7	110
1:408	PROTOCOL-ICMP Echo Reply	baja	192.168.1.19	4
129:12	stream5: TCP Small Segment Threshold Exceeded	media	192.168.1.7	52
1:29456	PROTOCOL-ICMP Unusual PING detected	media	192.168.1.7	4
1:366	PROTOCOL-ICMP PING Unix	baja	192.168.1.7	4
1:384	PROTOCOL-ICMP PING	baja	192.168.1.7	4
1:368	PROTOCOL-ICMP PING BSDtype	baja	192.168.1.7	4
1:40063	OS-LINUX Linux Kernel Challenge ACK provocation attempt	alta	192.168.1.7	100
129:15	stream5: Reset outside window	media	192.168.1.21	115
1:402	PROTOCOL-ICMP destination unreachable port unreachable packet detected	baja	192.168.1.21	67
1:12798	INDICATOR-SHELLCODE base64 x86 NOOP	alta	192.168.1.7	36
1:402	PROTOCOL-ICMP destination unreachable port unreachable packet detected	baja	192.168.1.7	4
1:40063	OS-LINUX Linux Kernel Challenge ACK provocation attempt	alta	192.168.1.21	400
1:44487	POLICY-OTHER SMBv1 protocol detection attempt	alta	192.168.1.19	1
1:44485	POLICY-OTHER SMBv1 protocol detection attempt	alta	192.168.1.19	1
1:44489	POLICY-OTHER SMBv1 protocol detection attempt	alta	192.168.1.19	1
1:40332	SERVER-WEBAPP Ruby on Rails Web Console remote code execution attempt	Alta	192.168.1.21	1
1:21268	SERVER-OTHER Oracle Java RMI services remote object execution attemp	media	192.168.1.19	1
1:22063	SERVER-WEBAPP PHP-CGI remote file include attempt	alta	192.168.1.19	1

1:34718	SERVER-WEBAPP ManageEngine Desktop Central FileUploadServlet directory traversal attempt	alta	192.168.1.21	2
1:36532	SERVER-OTHER Oracle Java JMX server insecure configuration remote code execution attempt	alta	192.168.1.21	2
1:32639	EXPLOIT-KIT Sweet Orange exploit kit jar file requested on defined port	alta	192.168.1.7	1
1:39995	POLICY-SOCIAL IRC server connection	alta	192.168.1.7	1
1:30375	INDICATOR-SHELLCODE Metasploit payload cmd*unix*reverse	alta	192.168.1.19	1
1:25106	MALWARE-BACKDOOR UnrealIRCd backdoor command execution attempt	alta	192.168.1.19	1
1:36256	SERVER-OTHER ElasticSearch information disclosure attempt	alta	192.168.1.21	3
1:33830	SERVER-OTHER ElasticSearch script remote code execution attempt	alta	192.168.1.21	1
1:18985	POLICY-OTHER CA ARCserve Axis2 default credential login attempt	media	192.168.1.21	1
1:5708	POLICY-OTHER web server file upload attempt	baja	192.168.1.21	1
1:19415	MALWARE-CNC vsFTPd 2.3.4 backdoor connection	alta	192.168.1.19	1

5.4. Elaboración de los perfiles de amenaza

Para la generación de los perfiles de amenaza se usó el escáner de vulnerabilidades Nessus home en su versión 7.1.3 Linux. La construcción de los perfiles de amenaza del servidor metasploitable3 se llevó a cabo usando los resultados de cuatro escaneos diferentes. En cada escaneo se usó la misma política de escaneo basada en la plantilla *Advanced Network Scan*. En cada escaneo se modificó fundamentalmente la familia de plugins usados. Es decir, en un caso particular se habilitaban todos los plugins seleccionados por defecto y en otros los plugin asociados con las máquinas Windows. El escaneo se ejecutó con credenciales (usuario, contraseña), lo que garantizaba un resultado más fino. Hay que anotar que se tuvieron en cuenta los cuatro ya que en unos se detectaban vulnerabilidades que en otros no, lo cual da a entender que es apropiado realizar más de un escaneo aun si se está usando la misma configuración para el escaneo, esto con miras a obtener resultados más precisos. En el caso del servidor metasploitable2 se usaron los resultados arrojados por un solo escaneo, ya que las variaciones con respecto a los otros dos escaneos realizados en términos de vulnerabilidades detectadas, era nula. El proceso para la construcción de los perfiles de amenaza se describe en el siguiente párrafo.

Los resultados de los escaneos se pueden exportar en formato Nessus que no es más que una extensión compatible con XML. En [101] se propone un script para convertir el formato de reporte de Nessus usando Python. A este script se le hicieron varios ajustes para cargar toda la información de los reportes almacenadas en una carpeta local. Para procesar la información de los reportes se creó un diccionario de diccionarios en Python, donde la clave del primer diccionario es la dirección IP del activo y la clave del segundo diccionario es el id de perfil. Para cada id de perfil se crea un diccionario con las claves mencionadas en el capítulo anterior (time, ip, port, protocol, plugin_id, etc.). A continuación se muestra un ejemplo de un perfil de amenaza luego de procesar los datos contenidos en varios reportes.

OS : Microsoft Windows Server 2008 R2 Standard Service Pack 1
protocol : tcp
severity : 4
cvss_base_score : 10.0
ip : 192.168.1.21
bid : ['87327']
cpe : cpe:/a:apache:struts
plugin_output :
Application : struts2-rest-showcase
Physical path : C:\Program Files\Apache Software Foundation\tomcat\apache-tomcat-8.0.33\webapps\struts2-rest-showcase\WEB-INF\lib\struts2-core-2.3.20.1.jar
Installed version : 2.3.20.1
Fixed version : 2.3.28.1

time : Tue Oct 9 14:17:29 2018
plugin_id : 90773
cve : ['CVE-2016-3081', 'CVE-2016-3082', 'CVE-2016-3087']
svc_name : cifs
port : 445

El siguiente es otro ejemplo de perfil de amenaza:

OS : Microsoft Windows Server 2008 R2 Standard Service Pack 1
protocol : tcp
severity : 0
ip : 192.168.1.21
cpe : cpe:/a:apache:tomcat
plugin_output :
URL : http://192.168.1.21:8282/
Version : 8.0.33
backported : 0
source : <title>Apache Tomcat/8.0.33

time : Tue Oct 9 14:17:29 2018
plugin_id : 39446
svc_name : www
port : 8282

En general, en Nessus hay unos plugin cuya categoría de severidad es informativa. Cuando se trata de la detección de un servicio la salida del plugin es similar a la que se muestra en la figura 12.

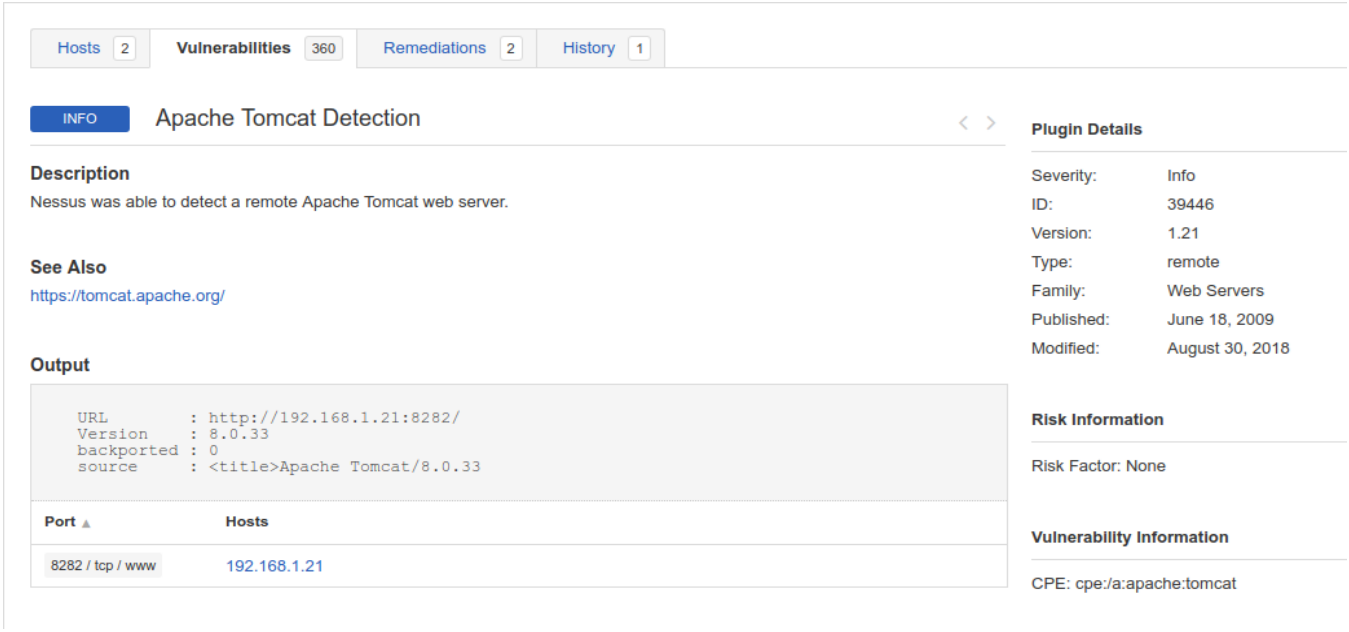

Figura 12. Salida de un plugin con severidad informativa vista en la interfaz web de Nessus.

Como se puede notar, en el perfil de amenaza no hay un campo versión del servicio, pero si se puede obtener a partir de la salida del plugin. En general, la estrategia que se implementa es la de por medio de expresiones regulares capturar las expresiones "*Version* :" e *"Installed version :"* y leer los datos que haya inmediatamente después hasta que se encuentre un salto de línea.

Puede que se presente la situación en la que Nessus no detecte una vulnerabilidad presente, pero que si detecte el servicio asociado a la vulnerabilidad y su versión. En este caso, al calcular el vector de correlación entre la alarma y el perfil de amenaza, las posiciones del vector asociadas al servicio y su versión serán 1, por lo que la alarma se clasificará como de interés. En este caso no se tuvo una coincidencia con la vulnerabilidad asociada con la alarma, pero sí con el servicio asociado con la vulnerabilidad, lo que hace de la alarma una alarma de interés.

Como se describió anteriormente, para el cálculo del score de prioridad se asumió que el activo aparece como un objetivo en la lista de activos comprometidos, por lo que se establece un valor de severidad por defecto igual a 3. Estos valores siguen la recomendación sugerida en la definición de la fórmula de prioridad [81,86]. Se asume que los dos servidores son activos críticos y que están modelados en la base de datos de activos corporativos, por lo que el valor por defecto del factor de confianza del modelo M, será 4. En cuanto a los otros dos factores de prioridad se establece que si se encuentra que el puerto está abierto y la posición 9 del vector de correlación es 1, es decir si se ha encontrado un coincidencia en la referencia a una vulnerabilidad, o si las posiciones 7 y 8 o 5 y 6 son 1 (si se encuentra una coincidencia en OS y versión o servicio y versión) se asume que el servicio es vulnerable por lo que se asigna un valor de 10 a MC y un valor de 10 a R. Si ninguno de los dos casos anteriores se cumple, pero se detecta el puerto abierto entonces se asigna un valor de 8 al factor MC y un valor de 5 al factor R. Por último, si el puerto se encuentra cerrado entonces al factor MC se le asigna el valor de 4 y al factor R el valor de cero.

Los criterios para determinar la validez de una alerta se basan en si el SO, el servicio y las versiones asociadas coinciden, además de la referencia de la vulnerabilidad.
En el siguiente capítulo se analizan los resultados obtenidos, pasando por cada vector de correlación de interés obtenido y mostrando la lista de alarmas priorizadas.

CAPÍTULO 6

Análisis de resultados y discusión

En este capítulo se analizan los resultados que arroja el escenario de pruebas y se hace una evaluación de la estrategia en cuanto a su capacidad para priorizar las alarmas asociadas con eventos de ataque.

6.1. Analizando los vectores de correlación obtenidos

Iniciaremos analizando las decisiones que tomó el algoritmo de similitud con alarmas asociadas con ataques que explotaban vulnerabilidades en los servidores.
El vector de correlación elegido por el algoritmo es el que se muestra en negrilla.

6.1.1. Servicios en Metasploitable3

Elasticsearch: 111100110 & 111100101

Perfil de alarma	Perfil de amenaza
time: 2018-10-24 22:41:11	time : Thu Oct 11 13:49:52 2018
ip: 192.168.1.21	ip : 192.168.1.21
port: 9200	port : 9200
protocol: tcp	protocol : tcp
OS-service: [(), ('elasticsearch', '1.1.1')]	OS : Microsoft Windows Server 2008 R2 Standard Service Pack 1
references: {'bugtraqid': [''], 'cve': ['2014-3120']}	
severity: high	severity : 0
alarm_id: 7219	cpe : cpe:/a:elasticsearch:elasticsearch
analyzer: snort	plugin_output : URL : http://192.168.1.21:9200/ Version : 1.1.1
signature_id: 1:36256	plugin_id : 109941
	svc_name : elasticsearch

Para el caso del servicio elasticsearch, luego de analizar los perfiles de amenaza se pudo evidenciar que en efecto la vulnerabilidad explotada se detectó, pero no se pudo extraer la información de la versión del servicio a partir de la salida del plugin que detectó la vulnerabilidad. No obstante, dado que esta alarma tuvo coincidencia con otro perfil de amenaza en servicio y versión además de los cuatro atributos iniciales, se clasifica como alarma de interés.

Apache Struts: 110100111 & 110100110

Perfil de alarma	Perfil de amenaza
time: 2018-10-17 17:04:55	time : Tue Oct 9 14:17:29 2018
ip: 192.168.1.21	ip : 192.168.1.21
port: 8282	port : 445

protocol: tcp	protocol : tcp
OS-service: [('windows', ''), ('apache struts', '2.3.20, 2.3.20.1, 2.3.24, 2.3.24.1, 2.3.28')]	OS : Microsoft Windows Server 2008 R2 Standard Service Pack 1
references: {'bugtraqid': [], 'cve': ['2018-11776', '2016-3087']}	cve : ['CVE-2018-11776']
severity: high	severity : 4
alarm_id: 5120	cpe : cpe:/a:apache:struts
analyzer: snort	plugin_output : Path : C:\Program Files\Apache Software Foundation\tomcat\apache-tomcat-8.0.33\webapps\struts2-rest-showcase\WEB-INF\lib\struts2-core-2.3.20.1.jar Installed version : 2.3.20.1 Fixed version : 2.3.35
signature_id: 1:39190	plugin_id : 112036 svc_name : cifs bid : ['105125'] cvss_base_score : 10.0

En el caso de apache struts, Nessus detectó la vulnerabilidad asociada con la alarma y al hacer la correlación de la alarma con el perfil correspondiente hubo coincidencia en servicio, versión y referencia. No obstante, hay que decir que esta vulnerabilidad se detectó consultando el puerto 445 mediante el protocolo cifs y no el 8282 asociado con la aplicación struts.

Apache Tomcat: **111100100**

Perfil de alarma	Perfil de amenaza
time: 2018-10-17 17:04:55	time : Tue Oct 9 14:17:29 2018
ip: 192.168.1.21	ip : 192.168.1.21
port: 8282	port : 445
protocol: tcp	protocol : tcp
OS-service: [('windows', ''), ('apache tomcat / hp operations manager', '3.0.*- 6.0.20 / 8.10')]	OS : Microsoft Windows Server 2008 R2 Standard Service Pack 1
references: {'bugtraqid': ['37086', '36954'], 'cve': ['2009-3843', '2009-3548']}	cve : ['CVE-2016-6816', 'CVE-2016-6817', 'CVE-2016-8735']
severity: high	severity : 3
alarm_id: 5124	cpe : cpe:/a:apache:tomcat
analyzer: snort	plugin_output : Installed version : 8.0.33 Fixed version : 8.0.39
signature_id: 1:17156	plugin_id : 95438 svc_name : www bid : ['94097', '94461', '94463'] Cvss3_base_score : 7.3

En el caso de Tomcat, Nessus no descubrió la vulnerabilidad relacionada con el ataque ejecutado. Aunque hubo coincidencia en el nombre de la aplicación, las versiones no coincidieron. Como ya se había mencionado, la alarma disparada por Snort asocia versiones de apache tomcat anteriores a la

versión instalada, por lo que en principio y de acuerdo a la lógica usada, se descartaría. Sin embargo, esta vulnerabilidad se instaló adrede en la versión de tomcat instalada en el servidor, que en primera instancia no sería vulnerable a este tipo de ataques (esta alarma se descarta para el análisis).

ManageEngine: 111100110 & 110100110

Perfil de alarma	Perfil de amenaza
time: 2018-10-24 22:41:11	time : Tue Oct 9 14:17:29 2018
ip: 192.168.1.21	ip : 192.168.1.21
port: 8020	port : 8020
protocol: tcp	protocol : tcp
OS-service: [(), ('manageengine desktop_central', '9.0')]	OS : Microsoft Windows Server 2008 R2 Standard Service Pack 1
references: {'bugtraqid': [], 'cve': ['2015-8249']}	cve :
severity: high	severity : 0
alarm_id: 7210	cpe : cpe:/a:zohocorp:manageengine_desktop_central
analyzer: snort	plugin_output : URL : http://192.168.1.21:8020/ Version : 9 build : 91084
signature_id: 1:34718	plugin_id : 71216 svc_name : www bid : Cvss3_base_score :

En el caso de Manageengine se reportó la alarma como alarma de interés dado que hubo coincidencia en servicio y versión no obstante que no se pudo establecer coincidencia en la referencia de la vulnerabilidad pues el escáner no la detectó.

Apache Axis2: 111100110 & 111100000

Perfil de alarma	Perfil de amenaza
time: 2018-10-24 22:41:11	time : Tue Oct 9 14:17:29 2018
ip: 192.168.1.21	ip : 192.168.1.21
port: 8282	port : 8282
protocol: tcp	protocol : tcp
OS-service: [(), ('apache axis2', '1.3 – 1.6')]	OS : Microsoft Windows Server 2008 R2 Standard Service Pack 1
references: {'bugtraqid': ['45625'], 'cve': ['2010-0219']}	cve :
severity: medium	severity : 0
alarm_id: 7222	Cpe: cpe:/a:apache:axis2
analyzer: snort	plugin_output : The following instance of Axis2 was detected on the remote host :

	Version : 1.6.0 URL : http://192.168.1.21:8282/axis2 Distribution : Servlet Services : Version
signature_id: 1:18985	plugin_id : 46739 svc_name : www bid : Cvss3_base_score :

En el caso de apache axis2 se reportó la alarma como alarma de interés dado que hubo coincidencia en servicio y versión no obstante que no se pudo establecer coincidencia en la referencia de la vulnerabilidad pues no se detectó.

Ruby on Rails: web console: 111100000 (múltiples)

Perfil de alarma	Perfil de amenaza
time: 2018-10-24 22:41:11	time : Thu Oct 11 13:49:52 2018
ip: 192.168.1.21	ip : 192.168.1.21
port: 3000	port : 3000
protocol: tcp	protocol : tcp
OS-service: [(), ('rubyonrails:web console', '2.1.2')]	OS : Microsoft Windows Server 2008 R2 Standard Service Pack 1
references: {'bugtraqid': [], 'cve': ['2015-3224']}	cve :
severity: high	severity : 0
alarm_id: 7206	Cpe:
analyzer: snort	plugin_output : The remote web server type is : WEBrick/1.3.1 (Ruby/2.3.3/2016-11-21)
signature_id: 1:40332	plugin_id : 10107 svc_name : bid : Cvss3_base_score :

A partir del perfil de amenaza se puede concluir que al no haber cpe asociado no es posible establecer la similitud en el campo servicio y por ende tampoco en la versión. Dado que no hay coincidencia en servicio y referencia, a esta alarma no se le asignará un score de prioridad alto, sin embargo, al presentar coincidencia en las primeras cuatro posiciones del vector, no será descartada por completo.

JMX: **111100000** (Múltiples)

Perfil de alarma	Perfil de amenaza
time: 2018-10-24 22:41:11	time : Tue Oct 9 14:17:29 2018
ip: 192.168.1.21	ip : 192.168.1.21
port: 1617	port : 1617
protocol: tcp	protocol : tcp
OS-service: [(), ('vmware vcenter server', '5.0-6.0')]	OS : Microsoft Windows Server 2008 R2 Standard Service Pack 1
references: {'bugtraqid': [], 'cve': ['2015-2342']}	cve :

severity: high	severity : 0
alarm_id: 7212	Cpe:
analyzer: snort	plugin_output : The Win32 process 'java.exe' is listening on this port (pid 2900).
signature_id: 1:36532	plugin_id : 34252
	svc_name : nimrod-agent?
	bid :
	Cvss3_base_score :

Este es el mismo caso que analizamos anteriormente. Nessus no detectó la vulnerabilidad explotada y tampoco descubrió el servicio asociado desde Snort. En este caso, la aplicación vmware no está instalada en la versión del servidor usado por lo que no se podría establecer una relación directa entre la alarma y la información del análisis de vulnerabilidad. El camino a seguir podría ser el de empezar a buscar por referencias asociadas a la vulnerabilidad explotada y ver si alguna de ellas fue detectada por el escáner de vulnerabilidades.

Apache Tomcat (Falsa alarma): **111100100** & 111100000

Perfil de alarma	Perfil de amenaza
time: 2018-10-24 22:41:09	time : Tue Oct 9 14:17:29 2018
ip: 192.168.1.21	ip : 192.168.1.21
port: 8282	port : 8282
protocol: tcp	protocol : tcp
OS-service: [(), ('apache tomcat', '4.1.0-6.0.16')]	OS : Microsoft Windows Server 2008 R2 Standard Service Pack 1
references: {'bugtraqid': ['35196'], 'cve': ['2009-0580']}	cve :
severity: medium	severity : 0
alarm_id: 7072	Cpe: cpe:/a:apache:tomcat
analyzer: snort	plugin_output : URL : http://192.168.1.21:8282/ Version : 8.0.33 backported : 0 source : <title>Apache Tomcat/8.0.33
signature_id: 1:18096	plugin_id : 39446
	svc_name : www
	bid :
	Cvss3_base_score :

En este caso particular se registra coincidencia en el nombre del servicio, pero no en la versión. Sin embargo, y como en los casos anteriores, esta alarma no será totalmente descartable a menos que el analista de seguridad lo decida, puesto que, al haber coincidencia entre 5 atributos del perfil de alarma con el perfil de amenazas, el score de prioridad no será cero.

Para tratar casos como el anterior se podrían crear dos categorías diferentes en los casos en que los vectores resultantes sean 111100000 & 111100100. El primer caso es en el que se chequea la versión del servicio, pero no coincide. Esta alarma sería una falsa alarma potencial. Otro es el caso cuando no hay información del servicio, por lo que no se puede determinar la medida de similitud. Este tipo

de alarmas podrían ser categorizadas en un estado de no determinado, por lo que habría que hacer un análisis adicional.

6.1.2. Servicios en Metasploitable2:

Unrealirc: 111100101

Perfil de alarma	Perfil de amenaza
time: 2018-10-24 22:41:11	time : Fri Oct 12 20:02:15 2018
ip: 192.168.1.19	ip : 192.168.1.19
port: 6667	port : 6667
protocol: tcp	protocol : tcp
OS-service: [(), ('unrealircd unrealircd', '3.2.8.1')]	OS : Linux Kernel 2.6 on Ubuntu 8.04 (gutsy)
references: {'bugtraqid': ['40820'], 'cve': ['2010-2075']}	cve : ['CVE-2010-2075']
severity: high	severity : 4
alarm_id: 7216	cpe: cpe:/a:unrealircd:unrealircd
analyzer: snort	plugin_output : The remote IRC server is running as : uid=0(root) gid=0(root)
signature_id: 1:25106	plugin_id : 46882 svc_name : irc bid : ['40820'] cvss_base_score : 10.0

La versión no se pudo extraer porque el output no tiene la expresión regular adecuada.

Samba: 111100000

Perfil de alarma	Perfil de amenaza
time: 2018-10-24 22:41:11	time : Fri Oct 12 20:02:15 2018
ip: 192.168.1.19	ip : 192.168.1.19
port: 139	port : 139
protocol: tcp	protocol : tcp
OS-service: [(), 'smbv1']	OS : Linux Kernel 2.6 on Ubuntu 8.04 (gutsy)
references: {'bugtraqid': [], 'cve': []}	cve :
severity: high	severity : 0
alarm_id: 7203	cpe:
analyzer: snort	plugin_output : An SMB server is running on this port.
signature_id: 1:44487	plugin_id : 11011 svc_name : smb bid : cvss_base_score :

En este caso, cuando se hizo la búsqueda de los sistemas afectados en la base de datos de Snort no se arrojaron resultados. En este tipo de escenarios no se pueden tomar decisiones por lo que es necesario que el analista haga un análisis más profundo acerca de la veracidad de la alarma.

RMI: 111100000

Perfil de alarma	Perfil de amenaza
time: 2018-10-24 22:41:11	time : Fri Oct 12 20:02:15 2018
ip: 192.168.1.19	ip : 192.168.1.19
port: 1099	port : 1099
protocol: tcp	protocol : tcp
OS-service: [(), ('vmware vcenter server', '5.0 – 6.0')]	OS : Linux Kernel 2.6 on Ubuntu 8.04 (gutsy)
references: {'bugtraqid': [], 'cve': ['2015-2342']}	cve :
severity: medium	severity : 0
alarm_id: 7207	cpe:
analyzer: snort	plugin_output : Port 1099/tcp was found to be open
signature_id: 1:21268	plugin_id : 11219
	svc_name : rmi_registry
	bid :
	cvss_base_score :

Este caso es similar al anteriormente descrito para Metasploitable3. Nessus no detectó la vulnerabilidad explotada y tampoco descubrió el servicio asociado desde Snort. En este caso, la aplicación vmware no está instalada en la versión del servidor usado por lo que no se podría establecer una relación directa entre la alarma y la información del análisis de vulnerabilidad.

PHP: 111100101

Perfil de alarma	Perfil de amenaza
time: 2018-10-24 22:41:11	time : Fri Oct 12 20:02:15 2018
ip: 192.168.1.19	ip : 192.168.1.19
port: 80	port : 80
protocol: tcp	protocol : tcp
OS-service: [(), ('php', '1.0- 5.4.2')]	OS : Linux Kernel 2.6 on Ubuntu 8.04 (gutsy)
references: {'bugtraqid': [], 'cve': ['2012-1823', '2012-2311', '2012-2335', '2012-2336']}	cve : ['CVE-2012-1823']
severity: high	severity : 3
alarm_id: 7208	cpe: cpe:/a:php:php
analyzer: snort	plugin_output : Version source : X-Powered-By: PHP/5.2.4-2ubuntu5.10, http://192.168.1.19/mutillidae/phpinfo.php Installed version : 5.2.4-2ubuntu5.10 Fixed version : 5.3.12 / 5.4.2
signature_id: 1:22063	plugin_id : 58988

svc_name : www
bid : ['53388']
cvss_base_score : 7.5

Aquí hubo coincidencia en la referencia. Podría incluso haber match en la versión, pero la cadena de texto extraída tiene la forma *5.2.4-2ubuntu5.10,* así que habría que mejorar el patrón de la expresión regular para que se pueda identificar la versión a partir de este tipo de cadenas de texto.

VSFTPD: 111100110

Perfil de alarma	Perfil de amenaza
time: 2018-10-24 22:41:11	time : Fri Oct 12 20:02:15 2018
ip: 192.168.1.19	ip : 192.168.1.19
port: 21	port : 21
protocol: tcp	protocol : tcp
OS-service: [(), ('vsftpd', '2.3.4')]	OS : Linux Kernel 2.6 on Ubuntu 8.04 (gutsy)
references: {'bugtraqid': ['48539'], 'cve': ['']}	cve :
severity: high	severity :
alarm_id: 7224	cpe: cpe:/a:php:php
analyzer: snort	plugin_output :
	Source : 220 (vsFTPd 2.3.4)
	Version : 2.3.4
signature_id: 1:19415	plugin_id : 52703
	svc_name : ftp
	bid :
	cvss_base_score :

En primera instancia no se encontró una coincidencia en el nombre del servicio dado que en el perfil de amenaza no existe el campo cpe. Debido a que el algoritmo de comparación de servicio y versión primero chequea si el nombre del servicio es igual, y en caso de que sea igual chequea la versión del servicio, la posición del vector asociada con la versión es 0, tal como si no se encontrara una coincidencia en la versión, siendo la misma.
Si se observa bien la salida del plugin, a partir de esta se podría extraer el nombre del servicio. Este tipo de salida es recurrente en varios plugins para varios tipos de servicio por lo que se podría modificar el algoritmo de extracción de nombre y versión del servicio para que acepte otros patrones de expresiones regulares.

Esta modificación se llevó a cabo y se pudo capturar el nombre del servicio a partir de la salida del plugin.

6.1.3. Analizando los vectores resultantes para las falsas alarmas

A parte del caso ya analizado correspondiente a la alarma con id 18096 donde se atacaba una vulnerabilidad presente en una versión de apache tomcat anterior a la versión instalada en el servidor, en breve se muestran los vectores de correlación obtenidos con otras falsas alarmas de interés:

Cisco IOS:	**110100000**
Paypal storefront:	**111100000**
IIS:	**111100000**

Apache Tomcat (signature_id: 17502): **111100100**

Apache Tomcat (signature_id: 12711): **111100100**

Para el caso de la vulnerabilidad de apache tomcat asociada con las referencias CVE-2009-3843 y CVE-2009-4189 se descubrió que al buscar información en las bases de datos de security-database y cvedetails aparecía como aplicación asociada HP Operations Manager. Al buscar en la base de datos de Snort, las versiones afectadas de apache tomcat van desde la 3.0 hasta 6.0.20. Sin embargo, la versión de apache tomcat instalada es la 8.0.33. De la información extraída de la base de datos de Snort se podría determinar que la versión instalada no es vulnerable a este tipo de ataques, sin embargo, el ataque se pudo ejecutar de forma exitosa. Por otro lado, se podría relacionar la alarma con la aplicación HP Operations Manager, sin embargo, esta no se encontró instalada en el servidor. La conclusión a la que se llega es que tal vez esta vulnerabilidad se instaló adrede en la versión de apache tomcat que tiene el servidor con fines experimentales. Con base en nuestra estrategia, al comparar las versiones de apache tomcat resultaría en una alarma que no coincide en versión del servicio y por tanto no sería clasificada como de alto interés. No obstante, la estrategia posibilita el análisis posterior de la alarma dado que no se descarta, a pesar de no asignársele una prioridad alta.

Teniendo en cuenta el caso anterior, surge a la vista la necesidad de incorporar herramientas adicionales al análisis para poder detectar casos como el anterior. Es aquí donde consideramos que la correlación con información de logs del sistema o de HIDS sería vital. No obstante que la vulnerabilidad se añadió adrede y que es probable que casos como estos no sean tan recurrentes, es importante tenerlos en cuenta y tomar las acciones pertinentes para resolverlos de manera exacta.

6.2. Alarmas priorizadas y análisis

Después de ser procesadas por la estrategia, cada una de las alertas recibió una puntuación en el rango de 4.5 a 1.0. Una puntuación más alta implica que la alerta es más relevante, mientras que una puntuación más baja implica que la alerta es irrelevante y puede ignorarse. En la tabla 22 se muestra un extracto de las alertas y puntuaciones asignadas por el prototipo.

Tabla 22. Alarmas priorizadas por la estrategia.

Id de firma	Firma	# Alarmas	Severidad (Snort)	IP	Puerto	Prioridad (Estrategia)
1:39190	SERVER-APACHE Apache Struts remote code execution attempt	2	alta	192.168.1.21	8282	4.5
1:10000002	VSFTPD Backdoor	8	alta	192.168.1.19	21	4.5
1:22063	SERVER-WEBAPP PHP-CGI remote file include attempt	1	alta	192.168.1.19	80	4.5
1:34718	SERVER-WEBAPP ManageEngine Desktop Central FileUploadServlet directory traversal attempt	2	alta	192.168.1.21	8020	4.5
1:25106	MALWARE-BACKDOOR UnrealIRCd backdoor command execution attempt	1	alta	192.168.1.19	6667	4.5

1:36256	SERVER-OTHER ElasticSearch information disclosure attempt	3	alta	192.168.1.21	9200	4.5
1:33830	SERVER-OTHER ElasticSearch script remote code execution attempt	1	alta	192.168.1.21	9200	4.5
1:19415	MALWARE-CNC vsFTPd 2.3.4 backdoor connection	1	alta	192.168.1.19	21	4.5
1:18985	POLICY-OTHER CA ARCserve Axis2 default credential login attempt	2	media	192.168.1.21	8282	3.4
1:40063	OS-LINUX Linux Kernel Challenge ACK provocation attempt	400	alta	192.168.1.21	8585	2.1
1:17156	SERVER-APACHE HP Performance Manager Apache Tomcat policy bypass attempt	4	alta	192.168.1.21	8282	2.1
1:30375	INDICATOR-SHELLCODE Metasploit payload cmd*unix*reverse	16	alta	192.168.1.19	6667	2.1
1:30383	INDICATOR-SHELLCODE Metasploit payload cmd_unix_reverse_ruby	3	alta	192.168.1.19	8787	2.1
1:17502	SERVER-APACHE Apache Tomcat UNIX platform directory traversal	10	alta	192.168.1.21	8282	2.1
1:32672	SERVER-OTHER Cisco ios ftp proxy overflow attempt	100	alta	192.168.1.21	21	2.1
1:2307	SERVER-WEBAPP PayPal Storefront remote file include attempt	100	alta	192.168.1.21	8282	2.1
1:44487	POLICY-OTHER SMBv1 protocol detection attempt	1	alta	192.168.1.19	139	2.1
1:44485	POLICY-OTHER SMBv1 protocol detection attempt	1	alta	192.168.1.19	139	2.1
1:44489	POLICY-OTHER SMBv1 protocol detection attempt	1	alta	192.168.1.19	139	2.1
1:40332	SERVER-WEBAPP Ruby on Rails Web Console remote code execution attempt	1	alta	192.168.1.21	3000	2.1
1:36532	SERVER-OTHER Oracle Java JMX server insecure configuration remote code execution attempt	2	alta	192.168.1.21	1617	2.1
129:15	stream5: Reset outside window	126	media	192.168.1.21/ 192.168.1.19	49671 /3309 2	1.5
1:29456	PROTOCOL-ICMP Unusual PING detected	25	media	192.168.1.21/ 192.168.1.19	p:icm p	1.5
1:31767	SERVER-OTHER MRLG fastping echo reply memory corruption attemp	4	media	192.168.1.19	P:icm p	1.5
128:4	Ssh: Protocol mismatch	1	media	192.168.1.19	22	1.5
1:12711	SERVER-APACHE Apache Tomcat WebDAV system tag remote file disclosure attempt	10	media	192.168.1.21	8282	1.5
1:1852	SERVER-WEBAPP robots.txt access	100	media	192.168.1.21	8282	1.5

1:977	SERVER-IIS .cnf access	10	media	192.168.1.21	8282	1.5
1:18096	SERVER-APACHE Apache Tomcat username enumeration attempt	100	media	192.168.1.21	8282	1.5
1:21268	SERVER-OTHER Oracle Java RMI services remote object execution attemp	1	media	192.168.1.19	1099	1.5
1:402	PROTOCOL-ICMP destination unreachable port unreachable packet detected	67	baja	192.168.1.21	P:icmp	1.0
1:366	PROTOCOL-ICMP PING Unix	25	baja	192.168.1.21/ 192.168.1.19	P:icmp	1.0
1:384	PROTOCOL-ICMP PING	25	baja	192.168.1.21/ 192.168.1.19	P:icmp	1.0
1:408	PROTOCOL-ICMP Echo Reply	4	baja	192.168.1.19	P:icmp	1.0
120:3	http_inspect: NO CONTENT-LENGTH OR TRANSFER-ENCODING IN HTTP RESPONSE	15	baja	192.168.1.21	80	1.0
123:3	frag3: Short fragment, possible DoS attempt	300	baja	192.168.1.21	unknown	1.0
123:8	frag3: Fragmentation overlap	284	baja	192.168.1.21	unknown	1.0
129:2	stream5: Data on SYN packet	100	baja	192.168.1.21	8282	1.0
1:5708	POLICY-OTHER web server file upload attempt	1	baja	192.168.1.21	8282	1.0

A partir de la tabla 22 se puede hacer una lectura del número de alarmas antes y después de ser procesadas por la estrategia. Con relación al número inicial de alarmas que fue de 2985, al pasar por la estrategia se redujo a 1858. Es decir, la reducción en la carga de alarmas fue de 37.8%. Lo anterior sólo considerando los filtros de IP. Si se estableciera un umbral de prioridad tal que las alarmas que estén por debajo del umbral se ignoraran, entonces la reducción en el volumen de alarmas sería de 77.9% (estableciendo un umbral de prioridad de 1.5)

La ventaja de este enfoque con relación al enfoque donde se usa un clasificador para predecir dos clases posibles, falso positivo o alarma verdadera, es que da la opción al analista de investigar aquellas alarmas con un score por debajo del máximo, pero que debido bien sea a que no se pudo obtener información de vulnerabilidades y/o información de servicios y versiones a partir de la alarma, no se ha clasificado con el máximo puntaje.
Por supuesto, teniendo en cuenta la arquitectura de la estrategia y las herramientas involucradas, se podría establecer la veracidad de la alarma generada involucrando los eventos registrados en los logs, como sería el caso de un HIDS.

Hay que decir que un gran porcentaje de las alarmas clasificadas como de alta prioridad fueron resultado de una coincidencia en términos de servicio y versión, más no de referencia a vulnerabilidad. Si bien es cierto que para los servidores usados hay vulnerabilidades asociadas con aplicaciones que no están instaladas en el servidor (en el caso de la vulnerabilidad asociada con una versión de tomcat inferior a la instalada) y que por otro lado hay alarmas para las cuales el sistema

afectado recogido de la base de datos de Snort no es el mismo que la aplicación que se está explotando, es necesario incorporar otros escáneres de vulnerabilidades con la intención de tratar de reducir los falsos positivos o falsos negativos generados por el VA.

6.3. Estableciendo una comparación con los antecedentes

Para poder establecer una comparación de la estrategia propuesta con los resultados de otros experimentos se requiere que el conjunto de datos usados sea el mismo. De hecho, esa es una de las anotaciones que hacen los autores en [8], la necesidad de tener un conjunto de datos estándar para poder establecer un análisis comparativo objetivo.
Si bien existen gran cantidad de conjuntos de datos usados para evaluar la capacidad de detección de IDS, estos no vienen integrados con la descripción necesaria de las vulnerabilidades de los equipos atacados.
Lo anterior fue un factor de peso para decidir elaborar un conjunto de datos, pequeño, por cierto, garantizando la correcta elaboración de los perfiles de amenaza a partir de los escáneres de vulnerabilidades. A pesar de ser un conjunto de datos relativamente pequeño, es superior al que se maneja en [75], además que la diversidad de los ataques [102] y la inclusión de tráfico de background lo proveen con los datos necesarios para hacer el análisis buscado y cumplir con los objetivos del proyecto.

Más allá de la restricción que pone el conjunto de datos usados, vamos a establecer una comparación relativa entre algunos trabajos consignados en el estado del arte y la estrategia propuesta.
Hay que empezar diciendo que en el caso de nuestra estrategia los datos se recogieron en bruto de la base de datos de Snort, en el caso de las alarmas y la información de las aplicaciones afectadas, y de los reportes de vulnerabilidades de Nessus en el caso de los perfiles de amenaza. Lo anterior quiere decir que no se añadió información adicional o se hizo un procesamiento manual más allá del ya mencionado sobre la aplicación y la versión asociada desde la base de datos de Snort. Por inferencia del analista se podría determinar tipo y versión de sistema operativo afectado, además de establecer relaciones entre referencias de vulnerabilidades del mismo tipo, en caso de que no haya coincidencia exacta con una, sí se relacione con otra del mismo tipo, y añadir esta información a las alarmas.
Sin embargo, a pesar de ser justificable e incluso válida la agregación de esta información, no iría de la mano con el concepto remarcado en nuestra estrategia que es el de automatizar la gestión de la seguridad. Es decir, al involucrar el componente humano en el ciclo, se estaría reduciendo el valor práctico a este tipo de implementaciones.

En las implementaciones realizadas en [75] y [1] no se especifica cómo se procesa la información obtenida a partir de las alarmas y cómo se relacionan los sistemas operativos y sus versiones, además de la construcción de los perfiles de amenaza. Ambos trabajos son de los mismos autores. En [75] usan una técnica de verificación donde se genera un vector de correlación con once posiciones. Este vector es el resultado de comparar las alarmas con perfiles de amenaza generados a partir de escaneos con Nessus y Nmap. La salida del módulo de verificación entra a un clasificador binario basado en redes neuronales que determina si la alarma es falsa o verdadera de acuerdo al vector generado.
En [1] los mismos autores mejoran la estrategia anterior, reduciendo la cantidad de posiciones del vector de correlación, y en lugar de hacer una clasificación binaria, usan una estrategia basada en teoría de juegos para determinar la veracidad de la alarma.
Hay que decir que los resultados obtenidos en ambos casos son muy buenos pues en [75] se reporta una exactitud del 97.25 % y una tasa de detección del 99.3% y en [1] se reporta una exactitud de 98.55% y tasa de detección de 91.87% para el conjunto de datos IITG. En nuestro caso se reportó una tasa de detección del 100% pues todos los ataques se procesaron, una precisión en la priorización del 100% y una tasa de falsos negativos en la precisión del 25%.

En la tabla 23 se registra la comparación de la lista de alarmas clasificadas con prioridad alta y media antes y después de aplicar la estrategia.
La tasa de falsos negativos en la priorización debe mejorarse. Esto se podría lograr añadiendo más información de contexto para tomar mejores decisiones.
Como se puede evidenciar a partir de la tabla 18, todas las alarmas referentes a eventos de ataque a las que no se les asignó el mayor score están ubicadas inmediatamente después de las alarmas clasificadas con prioridad alta, por lo que son alarmas factibles que se pueden examinar según determine el analista.

Tabla 23. Comparando la priorización de las alarmas antes y después de aplicar la estrategia.

	% Falsas alarmas	**% Alarmas verdaderas**	**Tasa de falsos negativos**
Antes	95.6	4.4	0
Después	0	100	25

Adicionalmente, queda claro que en la medida que se mejoren las firmas de Snort, se mejore la tasa de falsos negativos de los escáneres de vulnerabilidades y se incluyan los eventos de seguridad relacionados con otros sensores, esta tasa de falsos negativos se puede reducir a 0.
Con los resultados anteriores se demuestra que la estrategia reduce a 0 % la tasa de falsos positivos clasificados como de alta prioridad, aunque tiene una tasa de falsos negativos en la priorización del 25 %. Hay que decir que estas alarmas relacionadas con ataques a las que no se les asignó una prioridad alta, se les asignó un score medio, es decir que son alarmas factibles. Además, hay alarmas que indican post compromiso, es decir, cuando ya el atacante ha ganado acceso a la red protegida. Todas estas alarmas son clasificadas por Snort como de alta prioridad, sin embargo, la estrategia les asignó un puntaje de 2.1 (medio) puesto que no están asociadas directamente con una aplicación. A este tipo de alarmas se le podría dar un tratamiento especial, tal que puedan ser clasificadas como de alta prioridad. Mas allá de esto, la correcta detección de un ataque específico mediante la priorización de las alarmas relacionadas con las aplicaciones, evitaría o en caso de un ataque exitoso, establecería una asociación con la alarma de compromiso, por lo que sería identificable.

Conclusiones & trabajo futuro

Se demostró una estrategia híbrida para la priorización de alarmas usando técnicas de minería de alarmas. Luego de analizar los resultados de los vectores de correlación se determinó que no era buena idea hacer una clasificación binaria de falsa alarma o alarma verdadera. En lugar de ello, según el vector de correlación generado a partir de la medida de distancia entre las alarmas y los perfiles de amenaza, se calcula un puntaje de priorización que le dice al analista cuáles son las alarmas de interés. Se logró una reducción en el volumen de alertas del 77.9%, además de una tasa de falsas alarmas en la priorización del 0% y una tasa de precisión de priorización del 100%. Queda por mejorar la tasa de falsos negativos del 25%, que entre otras cosas se debió a los falsos negativos del escáner de vulnerabilidades y a la información incompleta de la base de datos de Snort. Como métodos de solución a este problema se propone la integración de más información de contexto considerando los logs del sistema e información proveniente de sensores heterogéneos.

Se plantea como trabajo futuro la implementación de la estrategia usando técnicas de grandes volúmenes de datos, pues como se demostró, la estrategia es altamente escalable pues los vectores de correlación se computan usando algoritmos basados en similitud. También se propone mejorar los algoritmos usados para la extracción de información de descripciones textuales, como lo fue en el caso de la extracción de información de sistemas afectados en la base de datos de Snort. Se propone usar técnicas semánticas para este fin.

Referencias

[1] B. Subba et al., False alarm reduction in signature-based IDS: game theory approach, Security Comm. Networks 2016; 9:4863–4881

[2] Loai Zomlot et al., Prioritizing intrusion analysis using dempster-shafer theory, In 4TH ACM Workshop on Artificial Intelligence and Security (AISec), Chicago, USA, Oct. 2011.

[3] S. Axelsson, "The base-rate fallacy and its implications for the difficulty of intrusion detection," in Proceedings of the 6th ACM Conference on Computer and Communications Security (CCS), 1999.

[4] Bob Violino, Security tools' effective hampered by false positives. [En línea] CSO Online, disponible en: https://www.csoonline.com/article/2998839/data-protection/security-tools-effectiveness-hampered-by-false-positives.html. [Visitado el 07 de abril de 2018]

[5] P.A. Porras, M.W. Fong, A. Valdes, A mission-impact-based approach to infosec alarm correlation, in: RAID 02', Proceedings of the 5th International Symposium on Recent Advances in Intrusion Detection, Lecture Notes in Computer Science, 2002, pp. 95–114.

[6] K. Alsubhi, E.A. Shaer, R. Boutaba, Alert prioritization in intrusion detection systems, in: NOMS '08: Proceedings of the 11th IEEE/IFIP Network Operations and Management Symposium, IEEE, 2008, pp. 33–40.

[7] Milan et al., Reducing False Alarms in Intrusion Detection Systems – A survey, International Research Journal of Engineering and Technology (IRJET), 2018.

[8] N. Hubballi et al., False alarm minimization techniques in signature-based intrusion detection systems: A survey, Computer Communications 49 (2014) 1–17.

[9] SIEM Prelude. [En línea] What is Prelude SIEM?, disponible en http://www.prelude-siem.com/. [Visitado el 10 de abril de 2018].

[10] Hurricane Labs. NSM and Intrusion Detection: Your Guide to Mastering IDS Rules and Alerts, disponible en: https://www.hurricanelabs.com/docs/idsguide.pdf [Visitado el 10 de mayo de 2018].

[11] Sitio oficial de Snort, disponible en https://www.snort.org/ [Visitado el 18 de abril de 2018].

[12] PCAP Next generation Dump File Format, sitio official, disponible en: https://www.tcpdump.org/pcap/pcap.html [Visitado el 25 de abril de 2018].

[13] Sitio oficial de Splunk, disponible en: https://www.splunk.com/ [Visitado el 25 de abril de 2018]

[14] Elastik Stack, sitio official, disponible en: https://www.elastic.co/elk-stack [Visitado el 25 de abril de 2018].

[15] Suricata IDS, sitio official, disponible en: https://suricata-ids.org/ [Visitado el 26 de abril de 2018].

[16] D. A. Bhosale and V. M. Mane. Comparative study and analysis of network intrusion detection tools. International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT). 2015. Pag. 312-315.

[17] Gartner peer insights. Reviews for Intrusion Detection and Prevention Systems, disponible en https://www.gartner.com/reviews/market/intrusion-prevention-systems [Visitado el 18 de mayo de 2018].

[18] Bro IDS. Sitio oficial, disponible en https://www.bro.org/ [Visitado el 13 de agosto de 2018].

[19] RFC 4765, The Intrusion Detection Message Exchange Format (IDMEF), disponible en: https://www.ietf.org/rfc/rfc4765.txt [Visitado el 13 de agosto de 2018].

[20] IETF, sitio oficial, disponible en: https://www.ietf.org/ [Visitado el 13 de agosto de 2018].

[21] J. Steinberger et al. How to Exchange Security Events? Overview and Evaluation of Formats and Protocols. In: 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM 2015), May 2015.

[22] B. Tung (1999), Common Intrusion Detection Framework, disponible en: http://gost.isi.edu/cidf/ [Visitado el 14 de agosto de 2018].

[23] Tools.ietf.org. RFC 5070 - The Incident Object Description Exchange Format. [En línea] Disponible en: https://tools.ietf.org/html/rfc5070 [Visitado el 14 de agosto de 2018].
[24] Common Announcement Interchange Format (CAIF). [En línea] Disponible en: http://www.caif.info/ [Visitado el 14 de agosto de 2018].
[25] NIST. Computer Security Resource Center (CSRC). Asset Reporting Format (ARF). [En línea] Disponible en: https://csrc.nist.gov/projects/security-content-automation-protocol/specifications/arf [Visitado el 14 de agosto de 2018].
[26] The CEE Board. Common Event Expression (CEE). MITRE, Junio 2008.
[27] J. Kohlrausch and S. U¨ belacker, X-ARF: A Reporting and Exchange Format for the Data Exchange of Netflow and Honeypot Data, 2011.
[28] C. Lonvick, The BSD Syslog Protocol, RFC 3164 (Informational), IETF, Aug. 2001.
[29] R. Gerhards, The Syslog Protocol, RFC 5424 (Proposed Standard), IETF, Mar. 2009.
[30] Intelligent Alert Clustering Model for Network Intrusion Analysis Int. J. Advance. Soft Comput. Appl., Vol. 1, No. 1, July 2009 ISSN 2074-8523.
[31] P0F V3. [En línea] Disponible en http://lcamtuf.coredump.cx/p0f3/# [Visitado el 15 de agosto de 2018].
[32] Prads (Home). [En línea] Disponible en https://gamelinux.github.io/prads/ [Visitado el 15 agosto de 2018].
[33] Symantec, Symantec Endpoint Protection. [En línea] Disponible en https://www.symantec.com/products/endpoint-protection [Visitado el 20 de agosto de 2018].
[34] ESET, ESET Endpoint Antivirus. [En línea] Disponible en https://www.eset.com/int/business/endpoint-security/windows-antivirus/ [Visitado el 20 de agosto de 2018].
[35] Sophos, Intercept X. [En línea] Disponible en https://www.sophos.com/en-us.aspx [Visitado el 20 de agosto de 2018].
[36] Crowdstrike, Falcon Prevent Next-gen Antivirus. [En línea] Disponible en https://www.crowdstrike.com/replaceav/ [Visitado el 20 de agosto de 2018.]
[37] Next Generation Antivirus, Carbon Black. [En línea] Disponible en https://www.carbonblack.com/products/solutions/use-case/next-generation-antivirus/ [Visitado el 20 de agosto de 2018].
[38] Cylance, Cylance Protect. [En línea] Disponible en https://www.cylance.com/en-us/platform/products/index.html [Visitado el 20 de agosto de 2018].
[39] ICSA. An Introduction to Intrusion Detection & Assessment. 1999.
[40] Ron Gula. Correlating IDS Alerts with Vulnerability Information. TENABLE Network Security. May 11, 2011 (Revision 4).
[41] Tenable. Nessus Vulnerability Scanner. Disponible en: https://docs.tenable.com/Nessus.htm [Visitado el 20 de agosto de 2018].
[42] Massicotte F, CoutureM,LabicheY. Context-based intrusion detection using snort, nessus and bugtraq databases. Proceedings of the Annual Conference on Privacy, Security and Trust, 2005; 1–12.
[43] Security Focus, BID. [En línea] Disponible en: https://www.securityfocus.com/bid [Visitado el 15 de septiembre de 2018].
[44] Common Vulnerabilities and Exposures (CVE). [En línea] Disponible en: https://cve.mitre.org/ [Visitado el 15 de septiembre de 2018].
[45] S. Bhatt, P. K. Manadhata and L. Zomlot. The Operational Role of Security Information and Event Management Systems. IEEE Security & Privacy. Volume: 12 , Issue: 5 , Sept.-Oct. 2014.
[46] Thomas Andrejak. Prelude-SIEM Documentation Release 4.0. Nov 15, 2017.
[47] El Mostapha Chakir et al., False Positives Reduction in Intrusion Detection Systems Using Alert Correlation and Data mining Techniques, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 5, Issue 5, 2015.

[48] Mirheidari S.A., Arshad S., Jalili R. (2013) Alert Correlation Algorithms: A Survey and Taxonomy. In: Wang G., Ray I., Feng D., Rajarajan M. (eds) Cyberspace Safety and Security. Lecture Notes in Computer Science, vol 8300. Springer, Cham.
[49] R.P. Goldman, W. Heimerdinger, S.A. Harp, C.W. Geib, V. Thomas, R.L. Carter, Information modeling for intrusion report aggregation, in: DISCEX 'II: Proceedings of the DARPA Information Survivability Conference and Exposition II, IEEE Computer Society, pp. 329–342.
[50] Sourcefire, Sourcefire ips the foundation of the sourcefire 3d system, White Paper, 2010.
[51] B. Morin, H. Debar, M. Ducassé, M2d2: a formal data model for ids alert correlation, in: RAID'02: Proceedings of the 5th International Conference on Recent Advances in ntrusion Detection, Lecture Notes in Computer Science, 2002, pp. 115–137.
[52] K. Julisch, M. Dacier, Mining intrusion detection alarms for actionable knowledge, in: SIGKDD '02: Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2002, pp. 366–375.
[53] K. Julisch, Using Root Cause Analysis to Handle Intrusion Detection Alarms. Ph.D. Thesis, IBM Zurich Research Laboratory, Switzerland, 2003.
[54] K. Julisch, Clustering intrusion detection alarms to support root cause analysis, ACM Trans. Inform. Syst. Sec. 6 (4) (2003) 443–471.
[55] K. Julisch, Mining alarm clusters to improve alarm handling efficiency, in: ACSAC '01: Proceedings of the 17th Annual Computer Security Applications Conference, IEEE, 2001, pp. 12–21.
[56] S.O. Al-Mamory, H. Zhang, New data mining technique to enhance ids alarms quality, J. Comp. Virol. 28 (2) (2010) 43–55.
[57] S.O. Al-Mamory, H. Zhang, Intrusion detection alarms reduction using root cause analysis and clustering, Comp. Commun. 32 (2) (2009) 419–430.
[58] Al-Mamory, S.O. & Zhang, H. J Comput Virol (2010) 6: 43. https://doi.org/10.1007/s11416-008-0104-2
[59] R. Perdisci, G. Giacinto, F. Roli, Alarm clustering for intrusion detection systems in computer networks, Eng. Appl. Artif. Intell. 19 (2006) 429–438.
[60] C. Dey, Reducing Ids False Positives Using Incremental Stream Clustering (isc) Algorithm, M.Sc. Thesis, Royal Institute of Technology, Sweden, 2009.
[61] N. Mansour, M.I. Chehab, A. Faour, ―Filtering intrusion detection alarms, Cluster Computing, Springer, 2010.
[62] Mohamed, A.B., Idris, N.B., Shanmugum, B.: Alert correlation framework using a novel clustering approach. Computer & Information Science (ICCIS), June 2012, Volume 1, pp 403-408
[63] T. Pietraszek, A. Tanner, Data mining and machine learning – towards reducing false positives in intrusion detection, Inform. Sec. Tech. Rep. 10 (3) (2005) 169–183.
[64] T. Pietraszek, Using adaptive alert classification to reduce false positives in intrusion detection, in: RAID'04: Proceedings of the 7th International Conference on Recent Advances in Intrusion Detection, Lecture Notes in Computer Science, 2004, pp. 102–124.
[65] R. Vaarandi, ―Real-time classification of IDS alerts with data mining techniques, in Proc. of MILCOM Conference, 2009.
[66] R. Vaarandi, K. Podins, ―Network IDS alert classification with frequent itemset mining and data clustering, IEEE Conference on Network and Service Management, 2010.
[67] Z. Tian et al., ―Reduction of false positives in intrusion detection via adaptive alert classifier, IEEE International Conference on Information and Automation, 2008.
[68] R Sadoddin et al., An incremental frequent structure mining framework for real-time alert correlation, Comp. Sec. 28 (2009) 153-173.
[69] M. Soleimani et al., Critical episode mining in intrusion detection alerts, in: Proceedings of the Communication Networks and Services Research Conference, IEEE Computer Society, 2008, pp. 157-164

[70] J. J. Treinen, R. Thurimella, Finding the needle: supression of false positives in large intrusion detection data sets, 2009 International Conference on Computational Science and Engineering, Vol. 2, pages 237-244.
[71] C. Mu, H. Huang, S. Tian, Intrusion detection alert verification based on multilevel fuzzy comprehensive evaluation, in: CIS '05: Proceedings of the 2nd International Conference on Computational Intelligence and Security.
[72] Spathoulas, Georgios P. and Sokratis K. Katsikas. "Reducing false positives in Intrusion detection systems." Computers & Security 29 (2010): 35-44.
[73] E. Allison Newcomb et al., Effective prioritization of network intrusion alerts to enhance situational awareness, 2016 IEEE Conference on Intelligence and Security Informatics (ISI).
[74] H. Fatma, M. Liman, A Two-Stage Process Based on Data Mining and Optimization to Identify False Positives and False Negatives Generated by Intrusion Detection Systems, 2015 11th International Conference on Computational Intelligence and Security (CIS).
[75] N. Hubballi, S. Biswas, S. Nandi, Network specific false alarm reduction in intrusion detection, Sec. Commun. Netw. 4 (2011) 1339–1349.
[76] F. Gagnon, F. Massicotte, B. Esfandiari, Using contextual information for ids alarm classification (extended abstract), in: DIMVA '09:Proceedings of the 6th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, Lecture Notes in Computer Science, 2009, pp. 147–156.
[77] T. Subbulakshmi et al., Real time classification and clustering of IDS alerts using machine learning algorithms, International Journal of Artificial Intelligence & Applications (IJAIA), Vol. 1, No.1, January 2010.
[78] S. Marchal et al. A Big Data Architecture for Large Scale Security Monitoring. 2014 IEEE International Congress on Big Data. 27 June-2 July 2014.
[79] J. Rees. Distributed multistage alert correlation architecture based on Hadoop. 2015 International Carnahan Conference on Security Technology (ICCST). 2015.
[80] A. Sima et al. A Hybrid Approach for Alarm Verification using Stream Processing, Machine Learning and Text Analytics. Published in Proceedings of the 21st International Conference on Extending Database Technology (EDBT), March 26-29, 2018.
[81] Fred Thiele, HP Arsight priority formula (2014). Disponible en: http://h41382.www4.hpe.com/gfs-shared/downloads-340.pdf [Visitado el 2 de noviembre de 2018].
[82] Prelude, Prelude-LML. [En línea] Disponible en: https://www.prelude-siem.org/projects/prelude/wiki/PreludeLml [Visitado el 2 de noviembre de 2018].
[83] Tenable network security. Nessus v2 File Format Last Revised: November 18, 2016. [En línea] Disponible en: https://static.tenable.com/documentation/nessus_v2_file_format.pdf [Visitado el 2 de septiembre de 2018].
[84] J. Haukeli. False positive reduction through IDS network awareness. Master Thesis University of Oslo. May 23, 2012.
[85] Tenable, Common Platform Enumeration (CPE) with Nessus. [En línea] Disponible en: https://www.tenable.com/blog/common-platform-enumeration-cpe-with-nessus [Visitado el 24 de septiembre de 2018].
[86] ArcSight ESM Threat Level Formula. Disponible en: https://community.softwaregrp.com/dcvta86296/attachments/dcvta86296/ArchiveDiscussionBoard/14499/1/Threat%20Level%20Formula.pdf [Visitado el 3 de septiembre de 2018].
[87] Manual Snort 3. Writing Snort Rules. Classtype. Disponible en: http://manual-snort-org.s3-website-us-east-1.amazonaws.com/node31.html [Visitado el 5 de noviembre de 2018].
[88] GNS3, Documentation. [En línea] Disponible en: http://docs.gns3.com/ [Visitado el 3 de septiembre de 2018].
[89] Prelude. Installing Snort. [En línea] Disponible en: https://www.prelude-siem.org/projects/prelude/wiki/InstallingAgentThirdpartySnort [Visitado el 4 de septiembre de 2018].

[90] Rapid7. Metasploit. [En línea] Disponible en: https://www.metasploit.com/ [Visitado el 5 de septiembre de 2018].
[91] Scot Morris. The Easiest Metasploit Guide You'll Ever Read. Copyyright 2018.
[92] Hacking Tutorials. Metasploit Tutorials. [En línea] Disponible en: https://www.hackingtutorials.org/metasploit-tutorials [Visitado el 7 de septiembre de 2018].
[93] Metasploitable. [En línea] Disponible en: http://www.pentestingexperts.com. [Visitado el 7 de septiembre de 2018].
[94] CBE-Pentester. Metasploitable-2. [En línea] Disponible en: http://tecraksa.github.io/blog/2015/03/15/metasploitable-2/ [Visitado el 2 de septiembre de 2018].
[95] Security Database. [En línea] Disponible en: https://www.security-database.com/ [Visitado el 8 de septiembre de 2018]
[96] Rapid7. Metasploitable3, Vulnerabilities. [En línea] Disponible en: https://github.com/rapid7/metasploitable3/wiki/Vulnerabilities [Visitado el 9 de septiembre de 2018].
[97] Another Infosec blog. Metasploitable 3 Walkthrough – Getting System. [En línea] Disponible en: https://two06.blogspot.com/2016/12/metasploitable-3-walkthrough.html. [Visitado el 9 de septiembre de 2018].
[98] Hacking Articles, Raj Chandel's blog. [En línea] Disponible en: http://www.hackingarticles.in/ [Visitado el 10 de septiembre de 2018].
[99] Hacking Tutorials. [En línea] Disponible en: https://www.hackingtutorials.org/ [Visitado el10 de septiembre de 2018].
[100] Bytes Over Bombs. [En línea] Disponible en: https://bytesoverbombs.io/ [Visitado el 10 de septiembre de 2018].
[101] Alexander V. Leonov. Parsing Nessus v2 XML reports with Python. [En línea] https://avleonov.com/2017/01/25/parsing-nessus-v2-xml-reports-with-python/ [Visitado el 20 de septiembre de 2018].
[102] McAfee Labs Threats Report (Marzo 2016). Disponible en https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-mar-2016.pdf [Visitado el 20 de septiembre de 2018].

.