## Modificaciones solicitadas 23/10/2020

Pequeña introducción sobre los riesgos a los que se exponen las redes modernas. Esto me llevará a la necesidad de justificar un SOC. Luego para desplegar un SOC/CSIRT, tomó como referencia el documento del SANS Inst. y esto me da lugar de forma natural, a presentar un SIEM.

CSIRT/SOC, su organización, la propuesta del SANS.

SIEM, sus funciones,etc…

Soluciones propietarias de SIEM, las versiones libres a las que dan soporte.

Soluciones open source para implementar un SIEM. Mencionar más de una para poder justificar la opción por la que voy a utilizar.

Los distintos tipos de CSIRT me van a permitir justificar la construcción de un CSIRT en la UNC.

### Hoja de ruta

1. Reordenar las partes
   1. comparar la estructura de lo solicitado con la que tenemos y cambiar lo necesario
   2. identificar los párrafos
   3. modificar las oraciones necesarias para mantener una coherencia en el texto
2. Incorporar la iteración 1 al marco teórico
3. Revisar los requerimientos y ordenarlos como se pide
4. Colocar los requerimientos después del marco teórico

# Descripción de Requerimientos

La idea es presentar requerimientos desde arriba hacia abajo. Desde el nivel de sistema hasta el nivel detallado.

## Requerimientos funcionales

* Envío de alertas sobre eventos destacables mediante servicios específicos: es necesario configurar servicios idealmente ya utilizados por la organización para enviar notificaciones sobre las detecciones de anomalías o potenciales incidentes por parte del sistema, para que los responsables del CSIRT y los departamentos que lo requieran, puedan tener conocimiento sobre las actividades detectadas y dar una respuesta adecuada.
* Analizar en busca puntos críticos de la red con mayor detalle al que se posee actualmente: es necesario que el nuevo sistema permita identificar los puntos críticos de la red y sus vulnerabilidades, así como el aporte de información sobre ellos que actualmente no es posible recoger con los medios de relevamiento y análisis de tráfico disponibles.
* Implementación de un sistema de notificaciones y pedidos para el manejo de incidentes: es necesario implementar un sistema de pedidos, o en su defecto adaptar el que la organización posea, para notificar al área responsable de la infraestructura o activo comprometido y que esta sea capaz de tomar las decisiones que considere apropiadas; tanto para mitigar las posibles consecuencias de un ataque como para fortalecer los activos. También será útil para constituir un registro sobre los pedidos realizados, obteniendo de esta manera un historial auditable.
* Automatización de respuestas a incidentes de seguridad: una vez detectado un evento y confirmado su clasificación como hostil, es necesario responder de la manera más rápida y eficiente posible. Además, muchos ataques consisten en una multitud de eventos que si dependieran de una respuesta individual a cada uno de ellos y realizada por un operador humano, sería imposible defenderse del tipo de ataques que consisten en miles de eventos lanzados al mismo tiempo, por ejemplo un intento de DDoS. Por otro lado, existen incidentes que se repiten periódicamente, como los de reconocimiento, en los cuales se puede aprovechar su naturaleza altamente repetitiva para desarrollar una respuesta automatizada y no distraer la atención de los operadores del sistema, permitiendo que estos se centren en tareas de mayor valor.
* Correlación de eventos: luego de detectar y eventualmente responder a incidentes, es importante contar con la capacidad de revisar las bases de datos y comparar los eventos actuales con los registros de incidentes del pasado, para establecer posibles correlaciones de firmas, tipo, objetivos u otros parámetros en tiempo real. Disponer de esta información es de gran importancia, ya que permite tener una visión y comprensión más profunda de la situación, lo que permitirá dar una respuesta mucho más efectiva.
* Visualización de los eventos en un tablero de mando: un aspecto fundamental de un sistema SIEM es la capacidad de integrar la información que envían los distintos sensores que están monitoreando y procesando tanto el tráfico en cualquier punto de la red (NIDS) como en los hosts (HIDS), de una manera concentrada, filtrada y ordenada para que los analistas puedan tener una visión clara y global de lo que está ocurriendo en todo momento, con la posibilidad de tener a disposición todos los detalles necesarios de los eventos que sean de interés. Además, debe ser posible tomar las decisiones haciendo uso de otros subsistemas del CSIRT, con sus respectivas acciones, desde el mismo tablero de mando.
* Normalizar eventos: el sistema debe ser capaz de filtrar los eventos provistos por los distintos sensores y normalizar la información para su posterior tratamiento, presentación y almacenamiento en las bases de datos respectivas.
* Almacenar eventos en una base de datos: el sistema debe almacenar la información normalizada de los eventos que procesa el SIEM, en una base de datos distribuida para posteriormente utilizar los registros en la correlación a demanda con otros eventos y proveer la capacidad de facilitar auditorías e investigaciones según sea requerido.
* Soporte para múltiples usuarios: el SIEM y sus subsistemas podrán ser operados por múltiples usuarios y con distintos niveles de privilegios, se trate de personal del SIEM, de otras áreas del CSIRT como de miembros de otras unidades de la organización que requieran tener acceso a registros puntuales.
* Recolectar registros de múltiples fuentes: Es un requerimiento fundamental que el sistema pueda almacenar los registros de todos los eventos o incidentes que sean necesarios y oportunamente seleccionados para su almacenamiento en las bases de datos

## Requerimientos no funcionales

* Escalabilidad de la solución: el desarrollo propuesto para el SIEM debe tener una arquitectura tal que permita escalar o redimensionar el sistema de acuerdo a la evolución de la infraestructura sobre la cual debe estar desplegado. Para ello, debe considerarse un diseño basado en componentes, modular y en contenedores, para desplegar inmediatamente en una nube propia de ser necesario. Es necesario que la solución desarrollada admita en un primer momento, la capacidad de monitorear enlaces de 1 Gbps de ancho de banda.
* Trazado de los diagramas topológicos de la red de la organización: es necesario realizar un relevamiento de las conexiones de la infraestructura de red interna de la organización central, las de sus dependencias y la red entre las unidades geográficamente distribuidas si las hubiera. Deben incluirse la topología de las conexiones de salida a Internet.
* Manipulación de las bases de datos resultantes para su agregación bajo demanda: es necesario adecuar la estructura de las bases de datos de manera que se encuentre de la manera más optimizada posible para recibir los logs y toda la información relacionada a los eventos. Las bases de datos deben cumplir condiciones de alta disponibilidad, particionado automático y diferentes niveles de persistencia en disco.
* Utilización de software libre: la solución propuesta debe contemplar el uso, configuración y desarrollo de software libre, cuyo código fuente pueda ser estudiado, modificado y utilizado libremente por la comunidad. Puede implementar licencias GPL, AGPL, BSD, Apache, GFDL, MIT y Creative Commons, según sea el caso de cada uno de los sistemas, subsistemas y componentes individuales.
* Uso de un sistema operativo de código abierto: es necesario que el sistema implementado se ejecute sobre un sistema operativo de código abierto, pertenecientes a las familias Linux o BSD.
* Instalación automatizada: desarrollar una secuencia de comandos que permita al sistema poder desplegarse de manera totalmente automatizada y sistemática en todos los servidores destinados por la organización. Para ello, se deben emplear lenguajes y metodologías que permitan el desarrollo de un proceso de instalación autónomo, la secuenciación de tareas para lograr una secuencia de etapas confiable, predecible y segura; transparentar el proceso de instalación y configuración inicial, minimizando el número de parámetros necesarios a configurar cuando el sistema entre en servicio. Además, el mecanismo empleado no debe contar con agentes para evitar la explotación de posibles vulnerabilidades en el proceso o requerir eventuales actualizaciones. Las técnicas y tecnologías empleadas deben permitir la orquestación de todas las configuraciones y mantenimiento que requiera el sistema y sus componentes, así como la entrega continua de mejoras y actualizaciones.
* Inventariar y clasificar los activos de la organización: es necesario investigar y realizar un relevamiento de los activos con los que cuentan las infraestructuras de red y de datos a fin de clasificarlos para poder organizar una estrategia que permita el despliegue y el uso más eficiente de los recursos del SIEM y del CSIRT en su conjunto, para garantizar el monitoreo más amplio posible, eliminar puntos oscuros o desprotegidos y obtener el proceso de respuesta más eficiente a eventos de seguridad de la información.

# Modificaciones solicitadas 28/10/2020

1. Figura 1: hacerla de nuevo y encontrar un diseño más atractivo
2. Resto de las figuras: controlar que tengan título
3. Reestructurar el marco teórico
4. Releer 2 veces el texto
5. Bibliografía: agregar citas y completar en la bibliografía
   * Cada vez que nombramos un producto, poner una referencia y en la bibliografía dejar el link al producto
6. Mantener los nombres y definiciones, no intercambiar términos equivalentes. Esto implica revisar nombres que hayan sido incluidos en gráficos, también los nombres de las tablas deben ser coherentes con la terminología utilizada en el texto.
7. Sacar el color a las tablas
8. Comentar en el marco teórico que elegimos Security Onion, pero sin dar los fundamentos.
9. Revisar todo el texto
   * Donde aparezcan referencias a los requerimientos, cambiarlas o eliminarlas ya que los requerimientos están al final.
   * Agregar info general de las otras herramientas libres (un párrafo o media carilla como máximo), ubicar al final ELK y security onion
   * .Revisar “Ámbitos del CSIRT”: presentarlo de una mejor manera y amagalmarlo o que dé lugar a “Estado de la ciberseguridad en Argentina”
     + “ciberseguridad en Argentina”: revisarlo y poner las figuras del comando de ciberdefensa y del csirt-nqn en una misma tabla.
     + Arreglar la “[figura N](https://docs.google.com/document/d/1T4uGL3amTLXEWKt_m22lFL2_wXn-OugTkkMTW6gscNw/edit?disco=AAAAHLphLSs)”: solo dejar el título y sacar la descripción hacia un párrafo

## Modificaciones 04/11/2020

### Corregir detalles del Marco Teórico

### Numerar los capítulos y subsecciones

### Corregir Descripción de Requerimientos.

1. Crear una introducción y tener en cuenta lo siguiente:
   1. Crear una introducción a los requerimientos funcionales. Está claro que el objetivo principal es selecciónar y desplegar un SIEM que tenga los RF descritos abajo…
   2. Inventariar y clasificar los activos de la organización: es necesario investigar y realizar un relevamiento de los activos con los que cuentan las infraestructuras de red y de datos a fin de clasificarlos.
   3. Trazado de los diagramas topológicos de la red de la organización: realizar un relevamiento de las conexiones de la infraestructura de red interna de la unidad central, las de sus dependencias y la red entre las unidades geográficamente distribuidas si las hubiera. Deben incluirse la topología de las conexiones de salida a Internet.
2. Definir los requerimientos funcionales y no funcionales, cambiar la redacción, usar infinitivos y el modo imperativo. **Ser sintácticos.**
3. Cerrar este capítulo con un orden de prioridades de los requerimientos sin distinción entre funcionales y no funcionales. Esto nos tiene que llevar a definir cómo van a estar ordenadas las Iteraciones que siguen
4. Ver si hay algún requerimiento (RF o RNF) que pertenezca al CSIRT y no al SIEM. En este caso, lo moveriamos a la intro

### Reorganizar, en la medida de lo resultante de las correcciones en la descripción de requerimientos, el contenido y orden en las Iteraciones.

### Pasar el informe a Latex (overleaf)

# Descripción de Requerimientos

Crear una introducción a los requerimientos funcionales. Está claro que el objetivo principal es seleccionar y desplegar un SIEM que tenga los RF descritos abajo…

Inventariar y clasificar los activos de la organización: es necesario investigar y realizar un relevamiento de los activos con los que cuentan las infraestructuras de red y de datos a fin de clasificarlos.

Trazado de los diagramas topológicos de la red de la organización: realizar un relevamiento de las conexiones de la infraestructura de red interna de la unidad central, las de sus dependencias y la red entre las unidades geográficamente distribuidas si las hubiera. Deben incluirse la topología de las conexiones de salida a Internet.

---

Con el objetivo de desplegar un sistema SIEM capaz de soportar los requerimientos funcionales y cumplimentar las condiciones que dictan los requerimientos no funcionales, es necesario definir el entorno en el que operará la plataforma.

Para esto se requiere, en primer lugar describir la topología de la red de la organización junto a lo que ello implica: realizar un relevamiento de las conexiones de la infraestructura de red interna de la unidad central, las de sus dependencias y la red entre las unidades geográficamente distribuidas si las hubiera. Deben incluirse la topología de las conexiones de salida a Internet.

En segundo lugar será necesario inventariar los activos de la organización: se requerirá una investigación y un relevamiento de los activos con los que cuentan las infraestructuras de red y de datos a fin de clasificarlos.

Las tareas de relevamiento anteriormente descritas proporcionarán un entendimiento acabado y profundo de la situación en la que se encuentra la infraestructura. Como resultado, será posible identificar puntos críticos a tener en consideración y como consecuencia, elegir la solución que mejor se ajuste a las necesidades de la organización.

* Implementar un sistema de autenticación y control de acceso de usuarios para el SIEM.

## Requerimientos funcionales del SIEM

* Recolectar y almacenar datos de incidentes de seguridad en la infraestructura de la red corporativa.
* Recolectar y almacenar información contextual y asociada a los activos vinculados al incidente.
* Visualizar las alertas en un tablero de mando.
* Implementar un sistema de envío de alertas de seguridad que notifique a los responsables de activos de información afectados.
* Definir un criterio para priorizar alertas.
* Implementar un sistema de correlación de alertas de seguridad.

## Requerimientos no funcionales

* La solución propuesta debe utilizar software libre
* El sistema operativo base debe ser tipo Unix y abierto.
* La arquitectura de la solución debe soportar la escalabilidad horizontal a demanda de la organización.
* Se requiere un despliegue automatizado de la solución.

-------------------------------------------------------------------------------------------------------

1. La solución propuesta debe utilizar software libre
2. El sistema operativo base debe ser tipo Unix y abierto.
3. Recolectar y almacenar datos de incidentes de seguridad de la infraestructura de la red corporativa.
4. Recolectar y almacenar información contextual y asociada a los activos vinculados al incidente.
5. La arquitectura de la solución debe soportar la escalabilidad horizontal del sistema.
6. Visualizar las alertas en un tablero de mando.
7. Implementar un sistema de notificación de alertas de seguridad a los responsables de activos de información.
8. Definir un criterio para priorizar alertas.
9. Implementar un sistema de correlación de alertas de seguridad.
10. Se requiere un despliegue automatizado de la solución.

En primer lugar, será seleccionada una plataforma cuyo código sea libre y abierto. Posteriormente será elegido un sistema operativo libre, tipo Unix, que sea compatible con la solución escogida. En segundo lugar, se adaptará la solución para recolectar y almacenar los datos pertinentes a los incidentes de seguridad que ocurren en la red corporativa, así como la información de contexto de los activos de información que se ven afectados. Se tendrá en cuenta el desarrollo de una arquitectura de despliegue que contemple la escalabilidad horizontal de la solución para adaptarse a las necesidades de la organización.

En tercer lugar, luego de haber recibido y almacenado los datos, se configurará la solución para visualizar la información disponible. Esto permitirá implementar un sistema de envío de alertas que notifiquen a los responsables de los activos de información que se viesen comprometidos. En consecuencia, se dispondrá de un sistema que permita correlacionar y filtrar alertas, en base a políticas a definir que incluyan las categorías de eventos y sus prioridades asociadas.

Finalmente se procederá a automatizar el proceso completo que comprende la implementación, despliegue y configuración de la solución junto a sus componentes asociados, mediante el uso de herramientas de automatización de tecnologías de la información.

Debe terminar con un texto que describa cómo continuará el trabajo en las siguientes iteraciones

# Versión de LáTeX

## Lista de cosas para hacer 06/11/2020

1. Estructurar el documento en overleaf de acuerdo al documento “Tesis Monitoreo de red v1.2” de google docs.
   1. Crear los módulos que falten y modificar los nombres de los que sean necesarios.
2. Crear carpetas de imágenes de acuerdo de acuerdo al capítulo
   1. Colocar las imágenes correspondientes con un nombre apropiado
   2. Crear indice de imagenes
   3. Las imágenes creadas con google sheet deben descargarse previamente en formato png / jpeg de ser necesario.
3. Bibliografía
   1. Pasar toda la bibliografía
4. Tablas
   1. Crear indice de tablas
   2. Pasar las tablas
5. Poblar con el contenido del documento “Tesis Monitoreo de red v1.2” de google docs.

Se quitó el título “Automatizando acciones: TheHiveHooks”

## Lista de cosas para hacer 09/11/2020

1. Hacer conclusión
2. Terminar glosario

## 

## 

## 

## Lista de cosas para hacer 11/11/2020

1. Enumerar los requerimientos funcionales y no funcionales
2. Realizar un análisis de riesgos para los requerimientos funcionales y no funcionales
   1. Hacer el cuadro del análisis de riesgo (nueva subsección) comparando requisitos vs soluciones disponibles.
   2. Hace una conclusión del cuadro.
   3. Hacer un cuadro con el orden de las prioridades de implementación de los requerimientos. Explicar las categorías de ese cuadro (baja prioridad, alta prioridad, etc)
   4. Como corolario de los puntos anteriores y usándolos para justificarnos, elegimos security onion.
   5. Arreglar la síntesis de cómo va a continuar el documento
3. Hablar de nuestra elección del sw (sonion) y todos sus componentes en el capítulo 4
   1. Implica unificar las iteraciones 1 y 2 en una sola.
   2. Mover la sección de “arquitectura de despliegue” al nuevo capítulo 5
4. El capítulo 6 sigue después del 4 porque después de explicar el sw que elegimos tenemos que analizar el ambiente de despliegue.
5. Arreglar la figura “4.6: Arquitectura de Despliegue”. Delimitar en el gráfico donde se encuentra la unidad central de la organización (para nosotros el PSI). Minuto 26:52
6. Agregar los requerimientos de hw originales en el capítulo de despliegue y terminar hablando del hw que disponíamos finalmente
   1. Contar los problemas que tuvimos y cómo los solucionamos (comportamiento anómalo de algunos componentes)
7. Agregar las versiones de todos los softwares y librerías usados.
8. Modificar la conclusión (minuto 1:17:38).
9. Describir el Test, usar las capturas de pantalla de la PSI (editadas).

**mike minuto 35:18:**

* Presentó los requerimientos.
* hago análisis de riesgos
* Opto por una solución.
* Seguir con los que hicimos (instalarlo dentro de las limitaciones que nos impuso la red de la universidad).
* Listo una vez que tenemos todo lo anterior tratar de darle cumplimiento a alguno de los requerimientos funcionales (esto sería el cap 5). Por ejemplo, ya tengo instalado el SIEM, lo tengo desplegado ¿como hago para recolectar datos de incidentes de seguridad? Se me ocurre hacer acciones ofensivas. Contar cuales fueron, como las generamos. ¿viste alertas de incidentes apenas levantaste la solución? ¿no tuviste ninguna dificultad? seguro que sí. Entonces en el proceso de recolección de datos de incidentes y de información contextual a lo mejor aparecieron problemas que los tuviste que solucionar, describirlos, contarlos

**mike minuto 56:10:**

¿como hace el fabricante para recomendarte el hw para el nodo master o un nodo forward? por el ancho de banda. Entonces hacer un cuadro “en base al ancho de banda el fabricante recomienda este hw para cada los tipos de nodo”

**mike minuto 01:06:30:**

en la iteración de despliegue (cap 6 en versión 1.5) debo terminar con alguna validación después del despliegue. ¿Cómo es válido? haciendo alguna prueba

Graylog y Sonion están iguales en requerimientos que cumplen. Creo que para tirar la para el lado de Sonion deberíamos agregar un requerimiento más, que esté relacionado al polimorfismo. Quizás eso lo podemos mencionar “por fuera” argumentando que nuestra tesis es una parte de algo más grande y que Sonion además de funcionar como SIEM puede funcionar como IDS.

## Lista de cosas para hacer 16/11/2020

### Capítulo 3

minuto 16:44

El documento queda definido en 3 iteraciones (16:58 minuto)

1. Iteración 1: Requerimientos Funcionales 1ero y 3ero
2. Iteración 2: Requerimientos Funcionales 2do, 4to y 6to
3. Iteración 3: Requerimiento Funcional 5to

Como consecuencia, eliminar el “capítulo 7: testing del sistema (7\_Iteracion\_4.tex)”.

(minuto 19:20) Explica que poner en el texto de “orden de implementación de prioridades”

### Capítulo 4

Comentar las imágenes de los casos de usos explicarlas un poco (ej Figura 4.14) (29:21 explica que poner en los párrafos de cada imagen)

Separar TheHive de Cortex: Deben estar en secciones (o subsecciones) separadas, con sus respectivos diagramas y capturas de pantalla.

Hablar un poco de OSSEC en la sección 4.7.1

minuto 37:15 dice cómo continúa el capítulo 5 (da pie a continuar el tema)

### Capítulo 5 (minuto 37:30)

minuto 38:53: reducir el tamaño de la letra de los títulos de los capítulos (Iteración N: blablabla)

En la sección 5.1 arrancar con texto y luego la imagen (43:07)

**Modificación de imagen**

46:41 conectar los cables al router

Dar nombre al switch, dar nombre a las vlans

Dejar abierto el enlace para otras vlans (vlan-j, vlan-m, minuto 51:05) y describir que son áreas de la red que dan acceso a otras dependencias desde el switch capa 3.

Dar nombre a los enlaces (54:30)

Decir de cuanto es el ancho de banda del enlace (56:11)

El datacenter tiene que estar dentro de la PSI (porque si).

Las PC de los analistas deben estar conectadas cada una a un puerto distinto

Desde el punto de vista del analista, sólo se ve el nodo master y thehive(57:38). Explicarlo

58:08 “5.2 configuración del ambiente de prueba” reemplazar por “selección del hardware”, saber que y como instale desde el nivel del SO hasta la solución con las configuraciones.

01:02:40 Descripción detallada del proceso de instalación de security onion como nodo master, forward. Lo mismo para thehive. Diferencias entre la instalación y configuración entre los nodos security onion si la hubiera.

Terminar el capítulo con una validación de la instalación.

01:00:19 Explicar 5.1 y 5.2 a qué corresponden esos requerimientos

01:03:56 Terminar con una validación de lo que se instaló

01:05:09 El trabajo no puede dar la impresión de instalar cosas

Agregar dibujo de paso intermedio

### 

### Modificación “Topología de la red (Capítulo 5)”

### Topologias de red

Durante el desarrollo de este proyecto, se evaluaron dos tipos de topologías de red. En primer lugar se ensayo con un tipo de topología monolítica que concentraba todos los componentes esenciales de Security Onion. Si bien se conocía que el rendimiento no era óptimo, se decidió comenzar por este punto para evaluar inicialmente al sistema.

Con el conocimiento adquirido sobre la naturaleza de Security Onion, se procedió a experimentar y finalmente desplegar en producción, versiones correspondientes a topologías distribuidas de la solución.

#### Topología Monolítica

Con motivo de familiarizarnos con Security Onion, se decidió desplegar este tipo de topología para facilitar la comprensión del funcionamiento de sus componentes. Se realizaron pruebas de los requerimientos funcionales 1 y 3: se comprobó la capacidad de recolectar y almacenar datos de incidentes de seguridad en la infraestructura de red de la organización, al enviar PCAPS e información previamente generadas de ataques que había sufrido la organización en el pasado. Se pudo observar estos eventos en un panel de mando, con lo que se verificó el cumplimiento del requisito funcional 3.

En la Figura 34n, se observa una típica topología de red en el cual se despliega la solución con una arquitectura monolítica. En el switch se encuentran conectadas dos terminales y el nodo monolítico de Security Onion. Una terminal simula ser la de un analista consultando y visualizando datos en Security Onion, mientras que la terminal restante envía logs conteniendo información contextual de los activos de la red.

|  |
| --- |
|  |
| Figura 34n: Topología monolítica de Security Onion |

El nodo monolítico de Security Onion dispone de dos conexiones. La correspondiente al puerto eth0 es la que permite la administración del sistema y la consulta de sus datos, mientras que la interfaz eth1 es la destina a monitorear el tráfico de una red. En nuestro experimento con esta arquitectura, todo el tráfico fue simulado usando los PCAPS mencionados anteriormente.

Este tipo de topología es ineficiente para el uso en entornos de producción, dado que la pila Elastic y los componentes de los sensores IDS demandan un uso intensivo de hardware. Por esta razón se procedió a analizar el despliegue de una topología distribuida.

### Topología distribuida

Después de comprobar el funcionamiento de los principales componentes de Security Onion y obtener la experiencia descrita en la sección anterior, se procedió a considerar el despliegue de una topología distribuida. Este tipo de topología presenta considerables ventajas respecto de su alternativa monolítica, fundamentalmente en términos de rendimiento del hardware y flexibilidad de adaptación, tal como se mencionó en el Capítulo 4 (referencia a la sección donde se tratan las arquitecturas).

En primer lugar, se desplegó una versión simplificada para verificar el funcionamiento de los componentes de los dos nodos que la componen (nodos Forward y Master) y la comunicación entre ellos.

|  |
| --- |
|  |
| Figura 35n: Versión elemental de una topología distribuida |

Se observa en la Figura 35n los componentes fundamentales de Security Onion para poder llevar a cabo esta topología: un nodo Forward conectado a la interfaz a monitorear y un nodo Master conectado al switch de la red interna. El nodo Master y el Forward se comunican mediante el enlace de administración del nodo Forward, que es el que está conectado al switch.

En este caso, como en el descrito en la sección anterior (referencia a la sección anterior), se utilizaron logs y tráficos de red obtenidos con anterioridad. El objetivo de esta experiencia fue comprobar el correcto funcionamiento de los dos nodos.

Adicionalmente, con el motivo de probar la generación de notificaciones de alertas, se comprobó el funcionamiento de ElastAlert, que forma parte de los componentes del nodo Master.

La experiencia consistió en enviar logs desde la terminal con Filebeat, para que posteriormente fueran filtrados. Para esta tarea se empleó logstash junto a un plugin llamado grok. Se buscó identificar campos de interés que estuvieran presentes en los logs y se hallaron direcciones IP. Se crearon alertas en ElastAlert que se activaron al detectar estas IP y se enviaron mensajes por un servidor de correo electrónico (propio de la organización) y aplicaciones (Telegram y Slack). Los resultados de este experimento cumplieron con el requerimiento funcional 4, que se trata en la iteración II (Capítulo 6(referencia del 6)).

Si bien Security Onion incluye gran parte de los componentes de un SIEM, es necesario otro sistema más que sea capaz de recolectar las alertas generadas en primera instancia por el nodo Master y manipular esta información para lograr un manejo eficaz de los incidentes. Este sistema es TheHive

Se observa en la Figura 36n el despliegue de TheHive junto a los nodos Forward y Master anteriormente mencionados. En esta ocasión, el servidor en el que se encuentra instalado TheHive está conectado a los otros nodos y los terminales mediante el switch.

TheHive recibe las alertas generadas en el nodo Master por ElastAlert y las introduce en un proceso de correlación de incidentes para proporcionar mayor información del mismo a los analistas del CSIRT. De esta manera, se completan los componentes del SIEM al ofrecer un manejo centralizado de los datos de incidentes de seguridad de la información.

|  |
| --- |
|  |
| Figura 36n: Versión simplificada de la topología distribuida con todos sus componentes |

Luego de la incorporación de TheHive, se volvió a repetir el experimento detallado anteriormente, esta vez con el objetivo de que TheHive recibiera las alertas generadas en el nodo Master mediante ElastAlert. A diferencia del experimento anterior, en el que el nodo Master recibía logs filtrados desde Filebeat, en esta ocasión se lanzó al nodo Forward un ataque simulado por su interfaz de monitoreo.

De esta manera, cuando el nodo Forward identificó el ataque, notificó al nodo Master del mismo. En el nodo Master, ElastAlert generó alertas que fueron enviadas al servidor de TheHive, quien las presentó a los analistas junto a información contextual que intentó correlacionar con información presente en su base de datos.

|  |
| --- |
|  |
| 38 n |

Posteriormente, se procedió a reemplazar la terminal que, con el objetivo de simular ataques, enviaba paquetes al nodo Forward de Security Onion. El enlace que estaba conectado a esta terminal fue reemplazado por otro unido a un puerto de un switch Capa 3. Este dispositivo estaba, a su vez, conectado con una dependencia de la organización. Se repitió el experimento pero en lugar de tener un ataque simulado, el tráfico monitoreado por el nodo Forward procedía del tráfico real de la dependencia que era redirigido mediante el switch capa 3.

Finalmente, se procedió a desplegar la configuración final de la topología de este proyecto. Esta consistió en agregar un segundo nodo Forward para monitorear el tráfico de una dependencia adicional, como se muestra en la figura 37n.

|  |
| --- |
|  |
| Figura 37n: |

La Figura 37n muestra la topología de prueba donde se instaló Security Onion. Se observan los proveedores de conexión a internet (ISP) y la conexión con el router de borde de la organización. Este último se conecta mediante un enlace gigabit en el puerto Gig0 al puerto Gig4 del switch 1 (capa 3).

El switch 1 es un dispositivo de red de capa 3, que conecta las dependencias entre sí y con el datacenter. Las conexiones mencionadas se implementan físicamente sobre un enlace gigabit de fibra óptica, pero virtualmente sobre redes tipo VLAN. De esta manera, por ejemplo, la dependencia 1 está conectada físicamente por un enlace gigabit a través de su puerto Gig1 de su router de borde, con el puerto Gig5 del switch capa 3. Desde el punto de vista lógico, este enlace pertenece a la VLAN 1 de la organización. La situación descrita es análoga para el resto de las dependencias de la Universidad.

Por otro lado, el switch 1 (en adelante SW 1), está conectado a los nodos Forward 1 y 2 de Security Onion. Como resultado, es posible reenviar el tráfico entre SW1 y las dependencias 1 y 2 hacia los puertos Gig0 y Gig2 de SW1 que conectan SW1 con los nodos Forward 1 y 2.

El SW1 está conectado con un switch 2 perteneciente al datacenter, al cual se conectan a su vez los terminales de los analistas, los nodos Master y Forward (1 y 2) de Security Onion y TheHive. Sobre esta topología final se realizaron las pruebas que se encuentran en las secciones posteriores.

### Verificación de los requerimientos funcionales 1 y 3

Luego de haber desplegado la topología distribuida que se observa en la Figura 5.11, se verificó el cumplimiento de los requerimientos funcionales 1 y 3. Para ello se llevó a cabo un reconocimiento de los puertos de una computadora ubicada en una dependencia de la organización. Se seleccionó esta computadora mediante un acuerdo con los responsables del área, para fines de evaluación de este proyecto.

Esta acción ofensiva consistió en un reconocimiento utilizando rutinas de NMAP(cita) (versión 7.80) configuradas a tal fin: nmap -T4 -A -v IP\_VICTIMA. Los parámetros utilizados fueron:

* -T4: para un escaneo intensivo (disminuye el tiempo de ejecución entre scripts)
* -A: habilita la detección del sistema operativo de la víctima y su versión, los scripts de escaneo y traceroute
* -v: habilita el modo “verboso”.
* IP\_VICTIMA: es la dirección IP objetivo de este reconocimiento.

En el recuadro rojo se puede observar la detección y visualización en Squert del reconocimiento realizado. El resto de la información que se aprecia en la Figura 38n corresponde a otros eventos que el sistema estaba detectando. Squert agrupa los eventos según la categoría a la que pertenecen y los clasifica mediante un indicador de colores (barra vertical a la derecha del contador de eventos de cada fila) según su prioridad de atención. Por defecto, Squert tiene su propia categoría de prioridades. Luego se observa la cantidad de eventos según su dirección, hora del último incidente ocurrido junto con su ID, una descripción de la firma del evento y el porcentaje de ocurrencia respecto al total de incidentes detectados.

|  |
| --- |
|  |
| Figura 38n: eventos visualizados en Squert |

En la figura 39n se observa la visualización de la detección en TheHive. Si bien pueden ser similares, Squert se limita a mostrar los ataques similares agrupados en la misma categoría de incidente al que pertenecen, mientras que TheHive recibe las alertas de ElastAlert y presenta los eventos de manera individual. Esto permite a los analistas crear casos con cada incidente, correlacionarse con otros ataques, etc, lo que conduce a una gestión integral de eventos de seguridad.

|  |
| --- |
|  |
| Figura 39n: detección en TheHive del ataque de reconocimiento |

Se verificó que las direcciones IP reportadas en el incidente por Squert y TheHive, coincidieran con las de la víctima y el atacante. Además mientras se estaba ejecutando el ataque, se hizo un seguimiento mediante wireshark(cita) filtrando las direcciones IP de origen y destino. Esto permitió corroborar el flujo de datos entre ambos.

Con esta prueba se verifica el cumplimiento de los requerimientos funcionales 1 y 3: “recolectar y almacenar datos de incidentes de seguridad en la infraestructura de la red corporativa ” y “visualizar las alertas en un tablero de mando”.

### Instalación y configuración de Security Onion

En primer lugar se creó una máquina virtual sobre el hipervisor VMWare ESX(i), con los requisitos definidos en la sección 5.1 para un nodo master.

Si sobre la máquina virtual se utilizó la ISO de Security Onion, la instalación es mediante un asistente integrado que solicita configurar la interfaz de red como enlace de administración (configuración de IP estática o dinámica, servidor DNS, etc).

En segundo lugar, el asistente pregunta si se desea configurar una interfaz para monitoreo. Este paso es importante ya que si lo que se está desplegando es un nodo master en una topología distribuida, hay que elegir la opción de no implementar una interfaz de monitoreo. Por el contrario, si se está desplegando un nodo Forward en una topología distribuida o un nodo Standalone en una topología monolítica, es necesario seleccionar la opción para configurar una interfaz para monitorear tráfico de red.

Luego de aceptar las modificaciones propuestas por el asistente, el sistema se reiniciará. Al iniciar nuevamente, será necesario volver a ejecutar el asistente de instalación, que nuevamente preguntará si se desea configurar las interfaces de red. Es necesario seleccionar la opción de saltar este paso, tras lo cual el asistente preguntará el modo de despliegue del sistema: evaluación o producción, como se ve en la figura 5.5

|  |
| --- |
|  |
| Figura 39n: selección de modo de despliegue |

Seleccionado el modo de despliegue, a continuación se debe elegir entre un nuevo despliegue de Security Onion o agregar la instalación actual a un entorno existente. Si el primer caso es seleccionado, se creará automáticamente un nodo Master, mientras que la opción restante configura un nodo Forward.

En el caso de elegir la primer opción, el asistente solicita el ingreso de datos para crear una cuenta de usuario y la elección de una configuración automática o manual para el resto de las opciones: persistencia de la base de datos mysql (por defecto 30 días), número de días de respaldo en caso de falla (por defecto 7 dias), selección de conjunto de reglas de detección (Emerging Threats Open o versiones de pago), motor de IDS (suricata o snort), habilitación de los sensores IDS (no recomendado para el nodo Master), SALT y activar o no la pila Elastic. Por último se debe seleccionar el almacenamiento de los logs: localmente o en un nodo de almacenamiento, así como el tamaño límite.

En el caso de elegir la opción de desplegar un nodo Forward, las configuraciones ofrecidas por el asistente son muy similares a las descritas para el nodo Master. Los cambios que se requieren son los datos relacionados a la configuración de red del enlace al nodo Master, por lo que es necesario que previamente exista este nodo. Finalmente, es necesario habilitar la conexión ssh e ingresar las credenciales del nodo Master, para la comunicación mediante un autotunel ssh entre ambos nodos.

### Guía del Capítulo 6

Para el requerimiento funcional RF2 explicar un poco más en detalle qué es Filebeat, como lo usamos. Contar lo de los logs de apache que teníamos. Hablar sobre la creación del filtro de Logstash utilizando el plugin grok. Cómo llegaron esos logs antes de crear el filtro. Cómo llegaron después de tener el filtro.

Hasta acá ya cumplimos con el RF2.

La motivación de crear filtros de Logstash para que los datos se separen en sus correspondientes campos, no fue solo a nivel “visual” para la presentación de los datos, sino que estos campos permiten realizar búsquedas para hacer correlaciones de con los datos utilizando ElastAlert.

Esto nos da pie para hablar de RF6 y RF4. Además podemos mencionar que RF6 (correlación) también la cumple en cierto punto Squert, Kibana y TheHive. El requerimiento RF4 se podría mencionar algo de los responders¿? sino lo pasamos de largo.

Finalmente el RNF3 se puede decir que fue mencionado y llevado a cabo en la iteración 1, obviamente explicamos un poco más en detalle esto.

### Guía del Capítulo 7

Mostrar algunas capturas de pantalla con los eventos priorizados y otra con no priorizados, los archivos de configuración si no los mencionamos antes.

Hablar un poco de todo lo que es necesario para que ansible funcione, lo que hicimos, etc

### Capítulo 6

#### Verificación de requerimientos funcionales 2: Recolección de información contextual

Con el objetivo de recibir y procesar información contextual de los activos que se ven afectados durante un incidente, fue necesario configurar el servidor Master de Security Onion con otros servicios que permitieran recibir información diferente de la provista por los sensores. Esta información refleja el estado de los servicios en un servidor, su conexión a la red, parámetros de uso de hardware como el nivel de ocupación de disco, temperatura, uso de la memoria RAM, los logs referidos al procesamiento de las peticiones que recibe un servidor, etc.

Una anomalía en los datos de esta información no es suficiente para confirmar un ataque, pero es útil para indicar que algo puede estar ocurriendo. En el caso de un ataque confirmado, estos datos representan evidencia forense y sirven para enriquecer modelos de amenazas complejas.

Se decidió enviar la información contextual disponible mediante Filebeat. Este es un servicio que permite enviar información a Logstash desde múltiples directorios. El proceso comienza cuando el servicio lee línea por línea los archivos de entrada y envía los logs de la misma manera a Logstash. Este los recibe en su puerto 5044 por defecto, posteriormente puede filtrarlos para separar los campos de los logs o bien almacenarlos sin filtrar.

Al principio enviamos los logs del estado de las aplicaciones de un servidor disponible en la organización, mediante Filebeat, a Logstash y sin un procesamiento posterior. Esto generó que ElastAlert no pudiera identificar información clave para realizar una correlación, como la que se encuentra en los campos de puertos y direcciones IP de origen y destino, estampas de tiempo, tipo de peticiones, etc.

Posteriormente utilizamos grok para filtrar los logs que estaba recibiendo Logstash antes de almacenarlos, de manera que la detección basada en correlaciones fue posible.

Como se observa en la Figura 6.1 y tal como se describió en la sección 5.3.2, realizamos el envío mediante Filebeat de logs de peticiones que fueron parte de ataques a activos de la organización en el pasado.

|  |
| --- |
|  |
| Figura 6.1: Despliegue distribuido de prueba |

Como se mencionó anteriormente, en primer lugar los logs se almacenaron sin filtrar, lo que ocasionó la pérdida de la capacidad de correlación. En la figura 6.2 se puede observar el almacenamiento de estos registros, con esto queda demostrado el cumplimiento del requerimiento funcional 2.

|  |
| --- |
| INSERTAR IMAGEN LOGS SIN PROCESAR |
| Figura 6.2: almacenamiento de logs sin procesar |

Posteriormente se repitió la experiencia pero esta vez los logs que se recibian en Logstash eran procesados utilizando grok. Grok es un plugin de la pila Elastic que permite filtrar datos sin ningún tipo de estructura y generar información estructurada capaz de ser consultada.

|  |
| --- |
|  |
| Figura 6.3: almacenamiento de logs procesados |

Logstash posee varios filtros para todos los tipos de entradas que soporta.

Es posible editar y administrar estos filtros utilizando archivos de configuración ubicados en la carpeta */etc/logstash/conf.d.available/*. Para este proyecto se eligió editar el filtro que procesa los datos provenientes de Filebeat, como se indicó anteriormente.

El proceso de creación de este filtro consistio en copiar el archivo de configuración de Filebeat: *9500\_output\_beats\_custom.conf* que se encuentra en el directorio mencionado anteriormente. El archivo fue copiado a la carpeta */etc/logstash/custom*, donde fue modificado para crear el filtro de grok. Este fue creado teniendo en cuenta el cuerpo de los mensajes provenientes del archivo de logs analizado, dado que se conocían los campos de interés que contenían los logs y que por lo tanto se deseaba extraer. Finalmente se reinicio el servicio de Logstash y se comprobó que el archivo *logstash.log* para verificar que no hubiera errores.

Es importante mencionar que la información resultante es enviada por Logstash a Elasticsearch, donde es consultada por ElastAlert. Este último componente es el encargado de correlacionar los campos de los logs y enviar notificaciones.

#### Verificación de requerimiento funcional 6 y 4: implementación de un sistema de correlación de alertas de seguridad y envío de alertas de seguridad

El envío de alertas de seguridad y la notificación a los responsables de los activos que se ven afectados, fue realizado en este trabajo mediante ElastAlert. Este componente fue seleccionado ya que permite correlacionar la información presente en los campos de logs guardados en elasticsearch, así como enviar notificaciones cuando se produce la detección de incidentes.

ElastAlert es un framework que detecta patrones en los datos consultados a Elasticsearch, como anomalías, picos, etc. Como se mencionó en la sección 4.5, basa su funcionamiento en dos componentes: reglas y alertas.

Las alertas consisten en mensajes que permiten notificar a un usuario final o a otro sistema, de que un evento de seguridad ha ocurrido. En este proyecto se decidió que las notificaciones fueran enviadas por correo electrónico a los responsables de un área de la organización.

Se escribieron archivos de configuración para cada tipo de evento, de manera tal que las reglas que disparan cada alerta lo hagan según el comportamiento y naturaleza del incidente. En las figuras 6.4 a 6.7 se observa el código desarrollado para el envío de alertas en ocasión de producirse un reconocimiento y escaneo de puertos. Las figuras corresponden al mismo archivo de configuración, pero cada una muestra una parte del código para facilitar la explicación del mismo.

En la Figura 6.4 se muestra la primer parte del código de configuración de una regla:

* es\_host: el nombre del host de la base de datos elasticsearch.
* es\_port: el número de puerto del host de elasticsearch mediante el cual ElastAlert se contactara con la base de datos.
* name: nombre de la regla. Debe ser único.
* type: especifica el tipo de regla. En este caso es del tipo “frecuency”, que determina que esta regla se activará si al menos se produce cierto número de eventos en determinado tiempo.
* num\_events: especifica la cantidad de eventos necesarios para activar este tipo de regla.
* timeframe: determina el marco temporal necesario para este tipo de reglas. tiene un sub parámetro que especifica la unidad temporal y la cantidad de estas. En este caso optamos por un marco temporal de un (1) minuto.

|  |
| --- |
|  |
| Figura 6.4: primera parte del código de configuración de una regla |

En la Figura 6.5 se muestran los campos de la regla para realizar la correlación de los datos:

* index: es el índice a consultar
* use\_strftime\_index: es una variable booleana que, en el caso de ser verdadera, da formato al índice utilizando “datetime\_strftime” para cada consulta. Este último método de Python crea un string representando el tiempo en un formato específico. Se utiliza en el caso de que la consulta realizada incluya varios días, con lo que los índices se concatenan utilizando “;” permitiendo una búsqueda más eficiente.
* filter: determina los filtros de elasticsearch utilizados para la consulta. Utiliza los subparametros “term” y “category” para indicar la clave a buscar. En este caso utilizamos “category: scan” para filtrar todos los eventos que pertenezcan a la categoría del reconocimiento de puertos.
* query\_term: permite agrupar las consultas, en este caso agrupamos las alertas resultantes por direcciones IP de origen y destino, así como todos los eventos que tengan el campo “alert”.
* realert: ignora las alertas repetidas en un cierto periodo de tiempo. En la práctica esto nos permitió evitar saturar con notificaciones a los responsables de los activos afectados, ya que el reconocimiento de puertos es un incidente extremadamente común y periodico, produciéndose muchas veces por día.

|  |
| --- |
|  |
| Figura 6.5: segunda parte del código de configuración de una regla |

En la Figura 6.6 se observan los campos utilizados para armar el cuerpo del mensaje y el asunto:

* doc\_type: especifica el tipo de documento a consultar en elasticsearch
* alert\_subject: permite personalizar el mensaje de una alerta agregando un pequeño resumen. En este proyecto lo utilizamos en todas las reglas, para que el “asunto” del correo electrónico contenga un mensaje de “Alerta: tipo-de-alerta”. En este caso el argumento ({0}) que figura es el primer campo (tipo de alerta) del objeto JSON que contiene la información del incidente.
* alert\_subject\_args: contiene el nombre del campo que será utilizado por alert\_subject
* alert\_text: contiene el mensaje de la notificación, se decidió presentar en una lista los parámetros más importantes de los incidentes. Estos incluyen el nombre de la alerta (que también está en el asunto del correo), timestamp indicando la fecha y hora de la detección del incidente, dirección IP de destino, el puerto de destino, dirección IP y puerto de origen, la interfaz del sensor que realizó la detección, información sobre la firma del incidente y un link para verlo en kibana.
* alert\_text\_type: especifica el formato del texto de la alerta, en este caso optamos por utilizar la sintaxis de formato standard de Python.
* alert\_text\_args: contiene los nombres de los campos cuyos valores serán utilizados para el contenido del mensaje.

|  |
| --- |
|  |
| Figura 6.6: tercera parte del código de configuración de una regla |

En la Figura 6.7 se observan los campos correspondientes al envío de la notificación:

* alert: determina el tipo de alerta implementada, es decir, el método de envío. ElastAlert puede enviar mensajes y notificaciones por varios medios que incluyen el correo electrónico, servicios y aplicaciones, como Telegram, JIRA, entre otros. En este proyecto se decidió utilizar el correo electrónico de la organización como medio de notificación de alertas.
* email: campo que contiene el correo electrónico de destino. Puede ser una dirección o una lista de direcciones de correos.
* from\_addr: este campo contiene el correo electrónico remitente que ElastAlert utilizará para enviar las notificaciones.
* smtp\_host: servidor SMTP del correo utilizado para enviar las notificaciones.
* smtp\_port: puerto utilizado por el servidor SMTP.
* generate\_kibana\_link: variable booleana que de estar activada genera un tablero temporal de Kibana y un link para acceder a el.
* use\_kibana4\_dashboard: link hacia un tablero de kibana (versión 4).

|  |
| --- |
|  |
| Figura 6.7: cuarta parte del código de configuración de una regla |

En la Figura 6.8 se observa un diagrama de secuencia del envío de una alerta.

La secuencia comienza cuando los logs que llegan a Logstash son filtrados y enviados de forma estructurada (JSON) al puerto 9200 de Elastisearch. ElastAlert, por otro lado, realiza consultas por logs a Elasticsearch. Cuando ElastAlert recibe una respuesta a su petición, procede a analizar los resultados en búsqueda de patrones que se puedan identificar en los logs recibidos. Independientemente de la coincidencia o no de algún patrón y la eventual notificación de alerta a los correspondientes responsables, ElastAlert vuelve a consultar a Elasticsearch por otros logs y procede a repetir el procedimiento descrito, mientras este activo como servicio. Si se produce la identificación de un patrón en los datos consultados, se dispara una alerta.

|  |
| --- |
|  |
| Figura 6.8: diagrama de secuencia del envío de una alerta |

Los patrones a buscar están definidos en las reglas habilitadas de ElastAlert y si se produce una coincidencia, se procede a enviar una notificación por los medios especificados en las reglas. Como se mencionó anteriormente, en este proyecto el medio elegido fue el correo electrónico de la organización. En la Figura 6.9 se observa el correo electrónico recibido cuando se disparó una alerta sobre un ataque de reconocimiento. Con esto damos por verificado los requerimientos funcionales 4 y 6, dado que para que la notificación haya sido enviada, previamente se tuvo que ejecutar una correlación, como se describió anteriormente.

|  |
| --- |
|  |
| Figura 6.9: Notificación recibida debido a una alerta por reconocimiento de puertos |

#### Verificación del requerimiento no funcional 3: Escalabilidad de la solución

Como se demostró en el capítulo 5, se desarrollaron distintas topologías de red, desde la monolítica hasta las variantes distribuidas. La topología distribuida admite un conjunto de varios nodos Forward y un nodo Master, lo que evidencia la capacidad de escalamiento horizontal de Security Onion. En este proyecto realizamos un despliegue de tres nodos Forward, sin embargo sólo dos nodos estuvieron activos la mayor parte del tiempo.

Dadas las restricciones de hardware con las que se contaba, no fue posible verificar el número máximo de nodos Forward que es posible administrar con un único nodo Master. Sin embargo consideramos cumplido el requerimiento no funcional 3, ya que se pudo demostrar el anexo de más de un nodo Forward sin repercusiones en el rendimiento general del sistema. La agregación de nuevos nodos Forward se realizó tiempo después de que el sistema se encontrará estable y funcionando, demostrando la viabilidad de escalar la solución en tiempo real sin detener su operación.

En la Figura 6.10 se observa una topología distribuida y simplificada en la que se encuentran desplegados dos nodos, que monitorean sendas dependencias.

|  |
| --- |
|  |
| Figura 6.10: Topología distribuida con dos nodos Forward |

## Capítulo 7

En Security Onion y otros sistemas, el elemento descriptor que identifica y procesa a cada definición de incidente en particular es la regla. Las reglas comprenden una serie de campos que describen con precisión la naturaleza de un incidente dado y por lo tanto, existen tantas reglas como amenazas en circulación.

Cuando un nuevo malware es descubierto por el equipo de algún CSIRT con la capacidad de investigación suficiente o reportado a un laboratorio apropiado para este fin, es posible realizar un estudio de sus características y una vez identificadas estas últimas, proceder a crear una regla y agregarla al repositorio correspondiente para que otros CSIRT actualicen sus IDS con esta nueva definición y así contar con un filtro (la regla) que permita detectar este malware. Las reglas tienen un conjunto de campos donde se detallan características del paquete y su contexto, tales como el puerto de origen y destino, protocolo empleado, dirección IP, etc y unos campos dedicados a la naturaleza del incidente (clasificación, mensaje, prioridad, etc). Algunos de estos campos son comunes a todas las reglas y permiten agruparlas para administrar eficientemente las alertas generadas cuando una regla coincide con la descripción de un incidente. Dado que estos campos también se pueden considerar observables, es posible utilizarlos por TheHive y Cortex para automatizar respuestas.

### Verificación del requerimiento funcional 5: Definición de criterio para priorizar alertas.

Como se indicó anteriormente, la estructura de las reglas consisten en dos partes bien definidas: un encabezado (header) que es obligatorio y un conjunto de campos opcionales. Dentro del header encontramos la acción (alerta, notificación, etc), el protocolo (tcp, udp), puertos de origen y destino, el sentido del evento (entrante o bidireccional) y las direcciones IP de origen y destino.

La segunda parte de las reglas incluye dos tipos de campos: los que describen la naturaleza del evento y aquellos que contienen información del paquete de datos. Dentro del primer grupo encontramos aquellos tales como msg (descripción del evento), sid (id de la firma), classtype (clasificación de reglas o alertas), priority (prioridad de la firma y/o alerta), target (especifica de qué lado está el objetivo, es decir puerto de origen y puerto de destino), entre otros. El segundo grupo contiene datos extraídos que provienen desde de la capa de red hasta la de aplicación de la pila OSI. Se pueden mencionar a los campos “GeoIP” (localización geográfica de la IP), “Fragbits” (presencia del bit de fragmentación), “ACK” (presencia del campo ACK en paquete TCP), “itype” (número del tipo de mensaje ICMP), “http.method” (tipo de método HTTP usado), entre otros.

|  |
| --- |
|  |
|  |

Como los campos están presentes en todas las reglas, es posible hacer uso de algunos de ellos para agrupar reglas que describen amenazas pertenecientes a un mismo grupo o categoría de malware, intentos de intrusión, reconocimiento, escalado de privilegios, etc y por lo tanto son útiles para gestionar los incidentes.

Es posible configurar esta gestión a través de un archivo que relaciona campos como categorías de eventos con prioridades de la alerta generada. Este archivo llamado “classification.config” se encuentra bajo el directorio que almacena las reglas descargadas desde diversas fuentes; en particular relaciona los campos “classtype” con “priority”, de manera tal que cualquier regla cuyo campo classtype contenga a los descritos en este archivo, generará una alerta con prioridad definida también en este. De esta manera, es posible administrar un enorme número de reglas agrupadas en un reducido grupo de categorías y modificar el nivel de prioridad que tendrá en el sistema las alertas que generan.

El objetivo de asignar distintos niveles de prioridad a las alertas generadas por los eventos que sucedan radica en la naturaleza de los eventos, su importancia y la gestión de la atención de los analistas del CSIRT. Esto se debe a las necesidades de optimizar el uso de los recursos técnicos y humanos del centro de respuesta a incidentes para cumplir de la manera más eficiente posible con los objetivos y políticas de la organización a la cual pertenece. De esta manera, la naturaleza de los incidentes determina su elegibilidad para una respuesta automatizada al tener en cuenta por un lado su estructura bien conocida y por el otro su alta tasa de repetición en un periodo determinado. En estos casos, sería inutil destinar valiosos recursos como la atención de un analista ya que conoce perfectamente la estructura del incidente y por lo tanto la respuesta apropiada o en aquellos casos en los que aún conocida su estructura, el incidente proviene en simultáneo de múltiples fuentes en muy poco tiempo, de manera que la capacidad humana de responder de a uno a la vez estaría tan sobrepasada que no sería efectiva. Estos son los casos de ataques de reconocimiento y los de denegación distribuida de servicio, entre otros.

De aproximadamente cuarenta y siete (47) categorías de incidentes disponibles por defecto, consideramos para el máximo nivel de prioridad a siete clasificaciones dado su nivel de ocurrencia y nivel de impacto para la organización.

* Web-application-attack: esta categoría engloba a un conjunto enorme de malware y ataques a nivel de capa de aplicación. Gusanos, ransomware, ataques de reconocimiento entre otras amenazas comparten esta categoría. Sobre el caso particular de los ataques de reconocimiento, se aplicaron filtros para separarlos de los demás ya mencionados.
* Unsuccessful User: intentos repetidos de ganar acceso en ciertos activos e infraestructura de la organización.
* Attempted-dos: intentos de ataque de denegación de servicio y su variante distribuida
* Known client side exploit attempt: intento de ejecución de exploits en el lado del cliente.
* Exploit Kit Activity Detected: detección de actividad de un kit de exploits
* A suspicious filename was detected: detección de nombres de archivos sospechosos
* Network Trojan: detección de un virus troyano de red.

Para verificar el cumplimiento del requerimiento funcional 5, se procedió a editar el archivo **classification.config** que se encuentra en el path **/etc/nsm/rules/.** En la Figura 7.2 se observa la detección de un incidente de reconocimiento en Squert. Este incidente pertenece a la categoría “attempted-recon” y en el archivo **classification.config** esta categoría tiene un nivel de prioridad 2, tal como se muestra en la figura (barra vertical naranja). Esto se puede observar con más detalle al analizar los campos de la alerta en Kibana, como se muestra en la Figura 7.3. Se remarcó el campo “priority”, donde se vio que efectivamente el valor de la prioridad es 2.

|  |
| --- |
|  |
| Figura 7.2:Incidente de reconocimiento en Squert |

|  |
| --- |
|  |
| Figura 7.3: Incidente de reconocimiento en Kibana |

Se modificó el archivo **classification.config** para elevar el nivel de prioridad de los eventos asociados a la categoría “attempted-recon”, que pasó del nivel 2 al nivel 1. Posteriormente se reiniciaron los sensores mediante el comando “so-sensor-restart” y se procedió a comprobar los resultados de la modificación. Se repitió el ataque de reconocimiento y se pudo observar en las Figuras 7.4 y 7.5 que Squert detectaba el ataque con prioridad 1 (barra vertical roja) y en Kibana se observó que el campo “priority” contenía el valor 1. Con esto se da por cumplido el requerimiento funcional 5.

|  |
| --- |
|  |
| Figura 7.4:Incidente de reconocimiento en Squert. Nivel 1 |

|  |
| --- |
|  |
| Figura 7.5: Incidente de reconocimiento en Kibana. Nivel 1 |

### Requerimiento no funcional 4: Se requiere un despliegue automatizado de la solución

Para el despliegue automatizado de la solución se utilizó Ansible 2.8.4. Ansible es una herramienta de automatización de administración de servidores, utilizada en la organización donde se desplegó este proyecto. Esta herramienta permite ahorrar tiempo y esfuerzo al automatizar tareas repetitivas, disminuir errores y aumentar la productividad general cuando se trata del despliegue e instalación de infraestructura básica, como instalar sistemas operativos, realizar la configuración inicial de estos, etc. Es por esta razón que la organización requirió el desarrollo de scripts de instalación de nuestra solución en Ansible.

En primer lugar, se desplegó una máquina virtual para probar el resultado de estos scripts. En esta máquina se instaló el sistema operativo Ubuntu Server 16.04 junto a los prerrequisitos para desplegar Ansible: Python en sus versiones 2.7 y 3.5, junto a su módulo pexpect (versión 4.7) y se configuraron las credenciales SSH entre esta máquina y el servidor desde el cual ordenamos el despliegue.

Posteriormente se desarrolló un archivo de configuración de Security Onion para su despliegue. Si bien este archivo incluye numerosos campos y variables, resaltan la configuración de red, el tipo de rol (Master o Forward) y en base a esto las interfaces de monitoreo o administración, así como configuraciones de almacenamiento de logs, etc.

Establecida la comunicación entre la máquina de destino y el servidor, se dio paso al inicio de la ejecución del script de Ansible. Para esto se ejecutó el comando “ansible-playbook -i hosts -l master so\_setup.yml --extra-var "target=sonionmaster" --ask-become-pass”. En este caso, el comando especifica el despliegue de un nodo Master, sus flags y variables son las siguientes:

* ansible-playbook: almacenan y ejecutan funciones de configuración, despliegue y orquestación de Ansible. Configura los pasos generales de un proceso de infraestructura IT y específica políticas para los sistemas remotos.
* - i hosts: flag que describe el inventario donde se encuentran los hosts. Esta varible “host” es un archivo que describe
* - l: opción utilizada junto al flag “-i”, selecciona un subconjunto de hosts del inventario.
* --extra-var: flag que configura variables adicionales como clave/valor o YAML/JSON. En nuestro ejemplo selecciona la opción sonionmaster que indica que se instalará un nodo Master de Security Onion
* --ask-become-pass: flag que solicita una contraseña de privilegio de usuario cuando sea necesario.

En el caso de la instalación de un nodo Forward, hubo que cambiar las opciones del comando que se mencionó anteriormente. El resultado fue:

“ansible-playbook -i hosts -l forward\_nodes so\_setup.yml --extra-var "target=sonionforward" --ask-become-pass”

En el caso de la instalación de TheHive y Cortex, el proceso es análogo al descrito en el párrafo anterior.

El comando para TheHive quedó de la siguiente manera:

ansible-playbook -i hosts -l thehive setup.yml --extra-var "target=thehiveuser" --ask-become-pass, mientras que en el caso de la instalación de Cortex, el comando resultante se muestra a continuación:

ansible-playbook -i hosts -l cortex setup.yml --extra-var "target=thehiveuser" --ask-become-pass

Entre las acciones que realizó Ansible durante la ejecución de los scripts, se destacaron las de descargar e instalar los paquetes de Security Onion, para luego configurar el sistema operativo utilizando el archivo de configuración mencionado anteriormente. En el caso de la instalación de un nodo Forward, durante la configuración del sistema se debió ingresar la clave ssh para conectarse con el nodo Master.

Finalmente se dio por satisfecho este requerimiento no funcional al comprobar que el sistema funcionaba correctamente.

## Lista de cosas para hacer 26/11/2020

Minuto 28:43 que esperaba mike encontrar en el capítulo 5:

Había una topología que nosotros encontramos en la universidad (topología sin nodos).

Yo entro y me encuentro con la topología de red que ya existe. Analizando esa topología de red yo me encuentro con que debería monitorear enlaces de dependencias que representan un ancho de banda de gigas (1 Gbit/s en nuestro caso). A partir de de ese dato de entrada, que es mi entrada para tomar la decisión de diseño ahí yo lo engancho y digo la recomendación del fabricante para poder monitorear enlaces de un 1Gbit/s (seria lo que ahora está en la sección 5.1) el hw recomendado para monitorear un enlace de 1 G/s es el que aparece por un lado en el Standalone y por otro lado en la arquitectura distribuida. Entonces el fabricante para una arquitectura Standalone me recomienda una cosa, para una arquitectura distribuida tal otra. Ok eso es lo que me recomienda el fabricante teniendo en cuenta la entrada de diseño (el criterio de diseño) de un enlace de 1G/b.

Bien ahora la pregunta es ¿Por que me quede con una arquitectura y no con la otra? ¿Por que me fui a una arquitectura distribuida y no a una standalone? Bueno eso sería lo que sigue. Eso sería decidir, bueno hice pruebas con que arquitectura me quedo y porque. La prueba no es configurar, la configuración es un detalle es operativo. Aparte de configurar desplegué una arquitectura haciendo una prueba con un nodo standalone me encuentro con que no se banca… (flashback!! volvemos un párrafo atrás minuto 35:38)

Una vez que yo digo a partir de la necesidad de monitorear un enlace de 1G/s el fabricante me recomienda para la arquitectura standalone tal hw para la distribuida tal otro ¿QUE SIGUE? ¿Tengo yo estas disponibilidad de hw? ¿Están disponibles en la UNC? Si el hw fuera infinito (como este informe) no habría problema de elegir cualquier alternativa. Pero no disponemos de todo el hw ¿entonces de qué hw dispongo? Dispongo de este hw (servidores impulsados por hamsters) lo describo. Entonces me encuentro en la disyuntiva de tener que elegir entre el despliegue monolítico y el descentralizado. Para eso pongo la topología de la UNC con el nodo Standalone hago las pruebas, veo que no resulta escalable. Debo encontrar las palabras correctas para decir que la arquitectura standalone no es la óptima. Entonces ahora sí hago la pruebas con la arquitectura distribuida, para eso nuevamente muestro la arquitectura de la UNC ahora con la arquitectura descentralizada, muestro los resultados de las prueba que hice. Entonces nuevamente la parte de configuración pasa a ser anecdótico lo que importa aca el es criterio con el que decido por una arquitectura y no por otra con las limitaciones que me impone la realidad que no es disponer siempre del hw que me recomienda el fabricante. Veo que pusieron unos gráficos de tráfico esa va a ser data que va a permitir definir el criterio para quedarme con una arquitectura y no con otra. Entonces terminó decidiendo por una arquitectura.

(break 39:38)

A lo mejor cuando voy describiendo la implementación de los nodos de prueba de la arquitectura Standalone hablo de en una pequeño título sobre la configuración de los nodos Standalone. Hago las pruebas llego a que las pruebas son estas, analisis de trafico, curvas, etc. Ahora hago las pruebas para la arquitectura distribuida hablo sobre la configuración de la arquitectura distribuida obtengo los resultados y después comparo ambos. Entonces digo me conviene una arquitectura distribuida y no standalone por tal y tal cosa. Concluyo el capítulo 5 diciendo que voy a implementar una arquitectura distribuida. Justifico ir por la arquitectura distribuida.

(minuto 43:14)

Estuvo mal arrancar hablando de la selección de hw sin haber empeza a hablar sobre la realidad en la que nos encontrábamos

(minuto 43:56)

Debo suponer que parto de una topología conocida de la organización (como la figura 5.13 sin el despliegue de los nodos). Sobre esta topología ¿cuales serian la opciones para desplegar el sistema de monitoreo? Bueno hay 2 una el la opción standalone y la otra es distribuirla. Para el caso de un nodo monolítico ¿donde lo conectaria para monitorear el tráfico? En el sw capa 3. Y en esa topología real ¿donde debería conectar yo una terminal para ver lo que está ocurriendo en ese sonion monolítico? Bueno como sonion cuenta con 2 interfaces una de monitoreo y otra de administración esta interfaz iría conectada al sw 2. Entonces para ver lo que está ocurriendo en el sonion monolítico se debe conectar una terminal a una de las interfaces del sw2. De esta manera sonion y la terminal estarían dentro de la misma vlan.

(minuto 47:46)

Ahora la pregunta es ¿Cómo escala la solución monolítica si quisiera monitorear N dependencias? No escala, porque deberíamos poner un nodo monolítico para cada dependencia y esto sería una demanda de hw muy grande. Si quiero monitorear un solo enlace de 1G/s no pasa nada. Ahora cuando quiero escalar eso y monitorear 25 enlaces de 1G/s las recomendación del fabricante pasa a ser un hw que está fuera de mi alcance. Esto se debe a que el nodo monolítico debe encargarse de la etapa IDS que hace uso intensivo de CPU y de la parte de almacenamiento y consultas de logs que también hace uso intensivo del hw.

(minuto 49:55)

Todo lo anterior sería el criterio que nos lleva a optar por una arquitectura distribuida.

(minuto 50:53)

Pongo en evidencia que el nodo monolítico funciona para un único enlace y no contradice lo que dice el fabricante porque para un 1G/s este hw anda. Pero esto no escala. Por que no escala? porque voy a querer sumar más enlaces de 1G/s, va a estar todo colapsado en un único hw (el monitoreo, el almacenamiento)

(minuto 51:35)

No debe haber un único servidor, tiene que estar replicado (necesidad de redundancia, prever un plan de contingencia). El plan de contingencia consiste en tener un servidor igual y disponible para conectar. Pero con N servidores, la solución (y su plan de redundancia) sería tan costoso que lo hace inviable, por lo tanto la solución no escalaría. Cerramos el capítulo 5 con la imagen 5.13, aunque también está bien como cerramos nosotros.

(minuto 55:44)

Sección 5.4: Verificación de requerimientos. Agregar la codificación de los requerimientos (RF1 Y RF3)

(minuto 58:04)

Cuando describa el ataque hacerlo con componentes del diagrama topológico. Hay que poner dirección IP fantasía por ejemplo “IP\_VICTIMA”.

(minuto 1:01:20)

Ponerle nombre a los routers de las organizaciones

(minuto 1:07:48)

Cambiar el enfoque de los ataques. Estos se hacen desde adentro de una dependencia hacia un activo dentro del datacenter, que puede ser un servidor víctima.

(minuto 01:09:51)

Que podría validar? que efectivamente se estan generando alertas. No es que de pronto solucione todo el problema, estoy teniendo la certeza que mi sistema desplegado en una topología de red real en una arquitectura distribuida, esta monitoreando, le estan llegando las alertas que estan generando en la dependencia, con la intención de llegar a los servidores de la prosecretaría.

(minuto 01:12:10)

Para el RF2 debemos enfocarnos en que queremos proteger. Entonces si nuestro activo utiliza apache sabemos que cualquier consulta que se realiza de un navegador genera un log de evento. Tengo que dejar en claro que esa información adicional me es útil para el caso de un ataque sobre un servidor apache versión tanto.

(minuto 01:15:07)

Debo enfocar mi protección en determinados servidores no en todos.

(minuto 1:15:12)

Lo primero que hice fue detectar un scaneo (el nmap) , lo debo dirigir sobre un servidor, que va a ser el que después le voy a recolectar información adicional (los logs).

(minuto 1:15:32)

Debo ir construyendo mi proceso de defensa capa sobre capa. Entonces al servidor le hago un ataque con el nmap, lo detecte. Bien ahora le recolecto los logs como muestro en el capitulo 6.

(minuto 1:16:56)

Debo proteger la información almacenada en servidores, las operaciones, las transacciones, la disponibilidad. No nos interesa proteger la red.

(minuto 1:18:54)

No estoy diciendo que lo haga pero estoy tratando de redefinir mis requerimientos funcionales: Yo necesito pensar en proteger activos de información, esos activos de información son servidores en una datacenter, en una granja de servidores en una organización importante. Entonces me voy a enfocar en atender la seguridad de esos servidores, entre ellos un apache (que es lo que voy a encontrar en cualquier organización)

(minuto 1:20:21)

Cerrar el capítulo 5 y 6 en esta dirección

(minuto 1:20:52)

Capítulo 7. Piensenlo, seamos consistentes en priorizar alertas, sobre un grupo de servidores que tiene cosas en común. Entonces sería priorizar alertas de servidores.

(minuto 1:22:41)

Capítulo 7. Reviso las reglas del IDS e intento ataques para activar determinadas alertas implementadas en las reglas del IDS. Por lo menos deberíamos implementar tres tipos de ataques, para saber si se están detectando y luego priorizarlos.

(minuto 1:24:02)

Ahora que yo detecto no solo uno, sino varios tipo de ataques y los puedo priorizar qué acciones debería poder generar que me automatice la defensa. Como no se llevo prácticamente las acciones describirlo a nivel teórico.

(minuto 1:32:26)

Capitulo 7 dice que voy a priorizar alertas, pero ¿por que voy a priorizar alertas? Debo justificar esta acción. El ¿para qué? debe ser el inicio del capítulo 7 porque eso es lo que dispara la necesidad de estudiar la priorización de alertas y los ajustes a las alertas.

(minuto 1:33:15)

Una solución el priorizar alertas, la otra solución al bloquear el tráfico

## Lista de cosas para hacer 26/11/2020

1. Hacer diagrama topográfico sin los nodos (topología de la UNC). (Hecho Topologia de la organizacion.png)
2. En base al diagrama anterior sabemos la topología que disponemos y que los enlaces que queremos monitorear son de 1G/s:
   1. Poner los requerimientos de hw recomendado por el fabricante para un nodo monolítico a la velocidad mencionada.
   2. Por otro lado, poner los requerimientos de hw recomendado por el fabricante para los nodos de una arquitectura distribuida también a una velocidad de 1G/s
3. Luego de describir los requerimientos recomendados por el fabricante para las dos arquitecturas a una velocidad de 1G/s. Siguen la pregunta ¿Tengo yo estas disponibilidad de hw? ¿Están disponibles en la UNC? Como no disponemos de suficiente hw estoy en la disyuntiva de elegir entre las 2 arquitecturas. Entonces:
   1. Pongo la topología de la UNC con el nodo Standalone (Hecho Topologia de despliegue monolitica.png)
   2. Hago las pruebas, análisis de tráfico, curvas, etc, sobre el nodo monolitico y veo que no resulta escalable.
      1. Cuando voy describiendo la implementación de los nodos de prueba de la arquitectura Standalone hablo de en un pequeño título sobre la configuración de los nodos Standalone
   3. Nuevamente muestro la arquitectura de la UNC ahora con la arquitectura descentralizada (Hecho Topología de despliegue descentralizada.png)
   4. Hago las pruebas con la arquitectura distribuida muestro los resultados de las prueba que hice
4. Bien ahora la pregunta es ¿Por que me quede con una arquitectura y no con la otra? ¿Por que me fui a una arquitectura distribuida y no a una standalone? Bueno eso sería lo que sigue. Eso sería decidir, bueno hice pruebas con que arquitectura me quedo y porque.
5. Luego de obtener los resultados de las pruebas que se llevaron a cabo para ambas arquitecturas, comparar dichos resultados. Entonces digo me conviene una arquitectura distribuida y no standalone por tal y tal cosa. Concluyo el capítulo 5 diciendo que voy a implementar una arquitectura distribuida. Justificó ir por la arquitectura distribuida.
6. Sección 5.4: Verificación de requerimientos. Agregar la codificación de los requerimientos (RF1 Y RF3).
7. Cuando describa el ataque hacerlo con componentes del diagrama topológico. Hay que poner dirección IP fantasía por ejemplo “IP\_VICTIMA”.
8. Ponerle nombre a los routers de las organizaciones
9. Capítulo 6. Cambiar el enfoque de los ataques. Estos se hacen desde adentro de una dependencia hacia un activo dentro del datacenter, que puede ser un servidor víctima. O el ataque puede venir desde afuera, desde más allá del ISP.
10. Capítulo 6. Para el RF2 debemos enfocarnos en que queremos proteger. Entonces si nuestro activo utiliza apache sabemos que cualquier consulta que se realiza de un navegador genera un log de evento. Tengo que dejar en claro que esa información adicional me es útil para el caso de un ataque sobre un servidor apache versión tanto.
11. Capítulo 6. Lo primero que hice fue detectar un scaneo (el nmap) , lo debo dirigir sobre un servidor, que va a ser el que después le voy a recolectar información adicional (los logs).
12. Capítulo 6. Proteger la información almacenada en servidores, operaciones, transacciones y disponibilidad. ¿redefinir requerimientos funcionales?
13. Capítulo 7 dice que voy a priorizar alertas, pero ¿por qué voy a priorizar alertas? Debo justificar esta acción. El ¿para qué? debe ser el inicio del capítulo 7 porque eso es lo que dispara la necesidad de estudiar la priorización de alertas y los ajustes a las alertas. (Sería mover lo que teníamos al principio).
14. Capítulo 7. Piensenlo, seamos consistentes en priorizar alertas, sobre un grupo de servidores que tiene cosas en común. Entonces sería priorizar alertas de servidores.
15. Capítulo 7. Reviso las reglas del IDS e intento ataques para activar determinadas alertas implementadas en las reglas del IDS. Por lo menos deberíamos implementar tres tipos de ataques, para saber si se están detectando y luego priorizarlos.
16. Capítulo 7. Ahora que yo detecto no solo uno, sino varios tipo de ataques y los puedo priorizar qué acciones debería poder generar que me automatice la defensa. Como no se llevó prácticamente las acciones describirlo a nivel teórico.
17. Capítulo 7: Una solución es priorizar alertas, la otra solución es bloquear el tráfico.
18. Capítulo 4.

# Iteración I: “Despliegue e instalación de Security Onion en un ambiente de prueba”

En este capítulo describimos la instalación de Security Onion en la topología de red de prueba. Se instalará un nodo Master, dos nodos Forward y un nodo de TheHive.

En este proyecto se trabajó primero sobre un ambiente de prueba y despues de producción. En ambos casos se utilizó un servidor central y un sistema operativo de virtualización sobre el que se crearon un conjunto de máquinas virtuales, cada una alojando un servidor con nodos Forward, Master y el correspondiente a TheHive - Cortex. Se utilizó de guía los componentes, el software y la arquitectura de conexión entre ellos, mencionados en la Descripción de Security Onion.

## Topología de la organización

El paso inicial consistió en relevar la topología de la organización, para ello se dispuso de información inicial que fue brindada por los responsables del sector de redes. Posteriormente se realizaron diagramas topológicos con el fin de evaluar la situación y contemplar distintas estrategias para el despliegue de este proyecto.

En la Figura 5.1 se observa un diagrama que representa la topología de la organización.

Se puede apreciar que las dependencias están conectadas (mediante sus routers de borde) entre sí y al mundo exterior mediante el Switch 1 de capa 3 que se encuentra en el datacenter. Se observan los servidores 1, 2 y N que representan los activos de información de la organización, estos están conectados mediante el Switch 2 al resto de la organización. Existen M switches para otras redes internas del data center, todos estos conectados al Switch 2. Este switch, por lo tanto, conecta las redes internas del datacenter y los activos de información con el Switch 3, siendo este último el switch troncal de la organización. Los enlaces anteriormente mencionados, incluidos los que conectan los routers de borde de las dependencias y el Switch 3, poseen un ancho de banda de 1 Gbps.

Se observan otros componentes de la infraestructura de la organización, encargados de la conexión exterior con el resto del mundo. Estos son el Firewall entre el Switch 1 y el Switch 5, los Routers de borde 1 y 2 junto al Switch 6 que se conecta a los proveedores ISP. Todos los enlaces entre estos componentes tienen como característica común un ancho de banda de 10 Gbps.

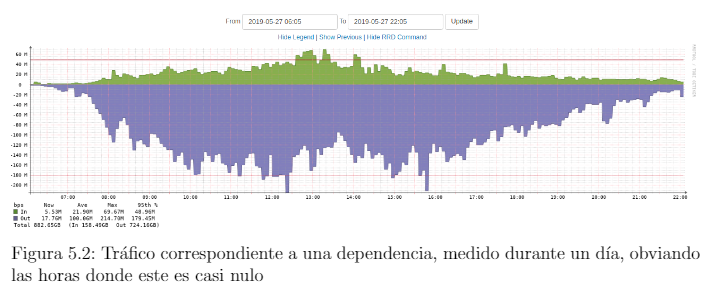
## 

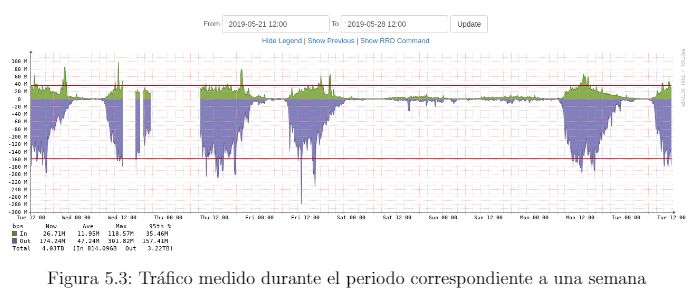
|  |
| --- |
|  |
| Figura 5.1: Topología de la Organización |

### Definición y configuración de las redes a observar

Se decidió monitorear dos dependencias en base a un análisis del ancho de banda de las dependencias existentes. Se seleccionaron las que mayor volumen de tráfico registraban en función de un registro histórico y mediciones propias realizadas a lo largo de una semana. Las dependencias seleccionadas tenían un enlace con ancho de banda de 1 Gbps cada una, con velocidades promedio consideradas como la suma entre entrada y salida, entre 11,95 y 47,24 Mbps respectivamente; con picos poco frecuentes de 300 Mbps de tráfico, que no se contemplaron en los requisitos de hardware, por lo tanto habrá una posible pérdida de paquetes en estos casos. Se realizó un “port mirroring” de los puertos del switch capa 3 a los que están conectados estas dependencias y se los conecto con los respectivos enlaces de monitoreo de sendos nodos Forward de Security Onion.

Las Figuras 5.3 y 5.2 fueron tomadas usando LibreNMS [cita] (versión 1.48), estas muestran el volumen del tráfico medido en dos periodos de tiempo distintos: Durante un día (exceptuando las horas en las que la actividad era mínima) y a lo largo de una semana. Si bien estos registros que se presentan a continuación corresponden a una sola de las dependencias, la restante tenía un comportamiento análogo.

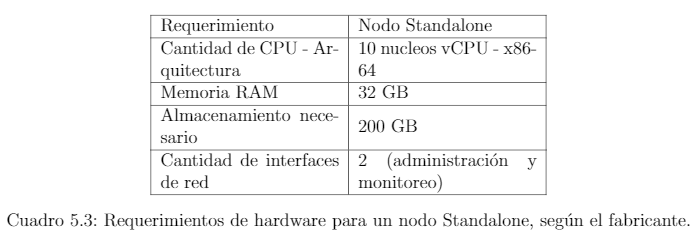




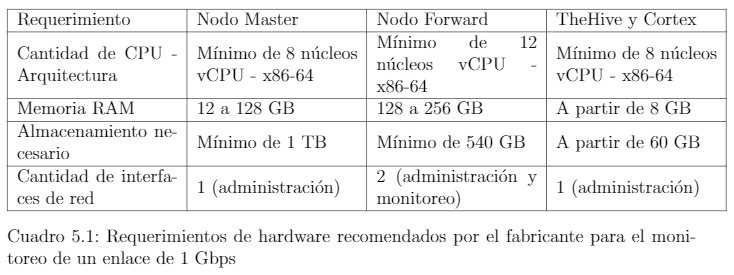
### Selección de hardware

El primer paso consistió en examinar los requisitos de hardware mínimos y recomendados por cada uno de los fabricantes de los sistemas y subsistemas elegidos. Se analizaron, por un lado, las demandas de tráfico de red en el ambiente de prueba y por el otro los requerimientos sobre los datos y capacidades que se esperan obtener del proyecto. En base al diagrama de la Figura 5.1, los datos referidos al tráfico y el ancho de banda de los enlaces a monitorear, se dimensionó el hardware necesario para el servidor central, teniendo en cuenta los requerimientos del fabricante y las limitaciones disponibles en la organización. Este servidor central albergó las correspondientes máquinas virtuales de este proyecto.

El fabricante especifica que, para un despliegue Monolítico o centralizado, los requerimientos de hardware de un nodo Standalone para monitorear un enlace de 1 Gbps, se muestran en el Cuadro 5.3:

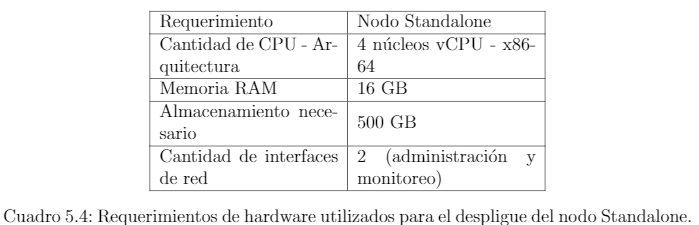


En el caso del despliegue de una topología Distribuida con enlaces de 1 Gbps, el fabricante recomienda los siguientes requerimientos de hardware para un entorno de producción. Estos figuran en el Cuadro 5.1.

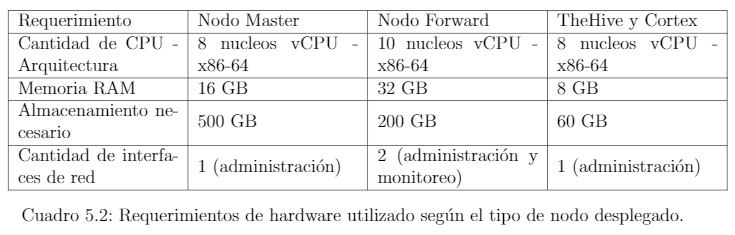


Debido a las restricciones en la disponibilidad del hardware, se realizó una implementación aproximada, teniendo en cuenta los recursos disponibles. A continuación se muestran los recursos de hardware utilizados, según se trate del despliegue de una topología Monolítica o de una Distribuida.

En nuestra evaluación del nodo Standalone, se utilizaron los recursos de hardware que figuran en el Cuadro 5.4.



En el caso de la topología Distribuida, el hardware que se utilizó para cada nodo se muestra en el Cuadro 5.2.



#### Características del hardware empleado

Si bien se dispuso de varios servidores a lo largo de este proyecto, cada uno con distinta capacidad de hardware, estas diferencias eran sólo cuantitativas, ya que las características del hardware para todos los servidores eran las mismas. A continuación, se detallan las más relevantes:

* En cuanto a los núcleos de CPU virtuales, su frecuencia era de 2.4 Ghz, basados en procesadores físicos Intel Xeon E5620.
* Las memorias RAM pertenecían a la tecnología DDR3 y su frecuencia de refresco era de 2400 MHz.
* Los discos utilizados fueron del tipo mecánico con una velocidad de transferencia para lectura de 196 MB/s y para escritura de 154 MB/s, con interfaz SATA III y 7200 rpm de velocidad de rotación.

## 

## Monitoreo de red

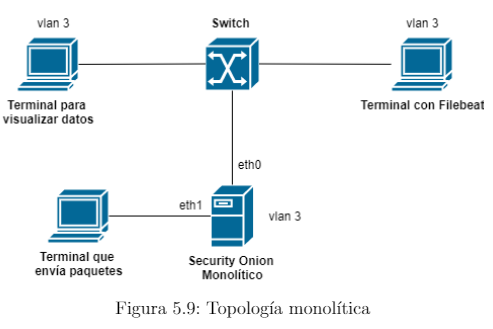
Durante el desarrollo de este proyecto, se evaluaron dos tipos de topologías de red. En primer lugar se ensayo con un tipo de topología monolítica que concentraba todos los componentes esenciales de Security Onion. Si bien se conocía que el rendimiento no era óptimo (sección 4.2.1), se decidió comenzar por este punto para evaluar inicialmente al sistema.

Con el conocimiento adquirido sobre la naturaleza de Security Onion, se procedió a experimentar y finalmente desplegar en producción, versiones correspondientes a topologías distribuidas de la solución.

### Topología Monolítica de Monitoreo

Con motivo de familiarizarnos con Security Onion, se decidió desplegar este tipo de topología para facilitar la comprensión del funcionamiento de sus componentes. Se realizaron pruebas de los requerimientos funcionales 1 y 3: se comprobó la capacidad de recolectar y almacenar datos de incidentes de seguridad en la infraestructura de red de la organización, al enviar PCAPS e información previamente generadas de ataques que había sufrido la organización en el pasado. Se pudo observar estos eventos en un panel de mando, con lo que se verificó el cumplimiento del requisito funcional 3.

En la Figura 5.9, se observa una típica topología de red en el cual se despliega la solución con una arquitectura monolítica. En el switch se encuentran conectadas dos terminales y el nodo monolítico de Security Onion. Una terminal simula ser la de un analista consultando y visualizando datos en Security Onion, mientras que la terminal restante envía logs mediante Filebeat [cite], conteniendo información contextual de los activos de la red. Esto permite cumplir el requerimiento funcional 2, que será tratado y demostrado en la Iteración II (Capítulo 6).



El nodo monolítico de Security Onion dispone de dos conexiones. La correspondiente al puerto Eth0 es la que permite la administración del sistema y la consulta de sus datos, mientras que la interfaz eth1 es la destina a monitorear el tráfico de una red. En nuestro experimento con esta arquitectura, todo el tráfico fue simulado usando los PCAPS mencionados anteriormente.

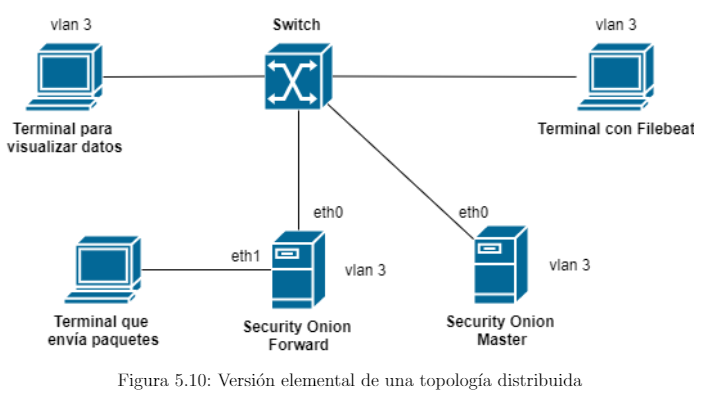
En la Figura 5.5 se observa la topología de la organización con un nodo Monolítico. Este tiene su interfaz de monitoreo conectada al Switch 3, donde recibe el tráfico reenviado que se produce entre la Dependencia 1 y el mencionado switch.

|  |
| --- |
|  |
| Figura 5.5 |

### Topología Distribuida de Monitoreo

Después de comprobar el funcionamiento de los principales componentes de Security Onion y obtener la experiencia descrita en la sección anterior, se procedió a considerar el despliegue de una topología distribuida. Este tipo de topología presenta considerables ventajas respecto de su alternativa monolítica, fundamentalmente en términos de rendimiento del hardware y flexibilidad de adaptación, tal como se mencionó en el Capítulo 4.

En primer lugar, se desplegó una versión simplificada para verificar el funcionamiento de los componentes de los dos nodos que la componen (nodos Forward y Master) y la comunicación entre ellos.



Se observa en la Figura 5.10 los componentes fundamentales de Security Onion para poder llevar a cabo esta topología: un nodo Forward conectado a la interfaz a monitorear y un nodo Master conectado al switch de la red interna. El nodo Master y el Forward se comunican mediante un enlace de administración (interfaz eth0 del nodo Forward), que es el que está conectado al Switch.

En este caso, como en el descrito en la sección anterior (referencia a la sección anterior), se utilizaron logs y tráficos de red obtenidos con anterioridad. El objetivo de esta experiencia fue comprobar el correcto funcionamiento de los dos nodos. Adicionalmente, con el motivo de probar la generación de notificaciones de alertas, se comprobó el funcionamiento de ElastAlert, que forma parte de los componentes del nodo Master.

La experiencia consistió en enviar logs desde la terminal con Filebeat, para que posteriormente fueran filtrados. Para esta tarea se empleó Logstash junto a un plugin llamado Grok. Se buscó identificar campos de interés que estuvieran presentes en los logs y se hallaron direcciones IP. Se crearon alertas en ElastAlert que se activaron al detectar estas IP y se enviaron mensajes por un servidor de correo electrónico (propio de la organización) y aplicaciones (Telegram y Slack). Los resultados de este experimento verificaron el RF4, que se trata en la iteración II (Capítulo 6).

Si bien Security Onion incluye gran parte de los componentes de un SIEM, es necesario un sistema adicional que sea capaz de recolectar las alertas generadas en primera instancia por el nodo Master y manipular esta información para lograr un manejo eficaz de los incidentes. Este sistema es TheHive.

TheHive recibe las alertas generadas en el nodo Master por ElastAlert y las introduce en un proceso de correlación de incidentes para proporcionar mayor información del mismo a los analistas del CSIRT. TheHive, Cortex y los componentes del nodo Master (Kibana y Squert) son los únicos componentes visibles con los que interactúan los analistas.

De esta manera, se completan los componentes del SIEM al ofrecer un manejo centralizado de los datos de incidentes de seguridad de la información. Cuando el nodo Forward identificó un ataque, notificó al nodo Master del mismo. En este último, ElastAlert generó alertas que fueron enviadas al servidor de TheHive, quien las presentó a los analistas junto a información contextual que intentó correlacionar con información presente en su base de datos.

Finalmente, se procedió a desplegar la configuración final de la topología de la solución en este proyecto. Esta consistió en agregar un segundo nodo Forward para monitorear el tráfico de una dependencia adicional, como se muestra en la Figura 5.13.

|  |
| --- |
|  |
| Figura 5.13: Despliegue de una topología distribuida de monitoreo |

La Figura 5.13 muestra la topología de prueba donde se instaló Security Onion. Se observan los proveedores de conexión a internet (ISP) y la conexión con el router de borde de la organización. Este último se conecta mediante un enlace gigabit en el puerto Gig0 al puerto Gig4 del switch 1 (capa 3). \par

El Switch 1 es un dispositivo de red de capa 3, que conecta las dependencias entre sí y con el datacenter. Las conexiones mencionadas se implementan físicamente sobre un enlace gigabit de fibra óptica, pero virtualmente sobre redes tipo VLAN. De esta manera, por ejemplo, la Dependencia 1 está conectada físicamente por un enlace gigabit a través de su puerto Gig1 de su router de borde, con el puerto Gig5 del Switch 1 capa 3. Desde el punto de vista lógico, este enlace pertenece a la VLAN 1 de la organización. La situación descrita es análoga para el resto de las dependencias de la Universidad. \par

Por otro lado, el Switch 1 (en adelante SW 1), está conectado a los nodos Forward 1 y 2 de Security Onion. Como resultado, es posible reenviar el tráfico entre SW1 y las dependencias 1 y 2 hacia los puertos Gig0 y Gig2 de SW1 que conectan SW1 con los nodos Forward 1 y 2. \par

SW1 está conectado con un switch 2 perteneciente al datacenter, al cual se conectan a su vez los terminales de los analistas, los nodos Master y Forward (1 y 2) de Security Onion y TheHive. Esta topología permite desplegar una arquitectura distribuida de Security Onion, que fue la elegida para este proyecto.

### Verificación de RF1 y RF3

Luego de haber desplegado la topología distribuida que se observa de la Figura 5.8, se verificó el cumplimiento de los requerimientos RF1 y RF3. Para ello se llevó a cabo una acción ofensiva sobre un servidor ubicado en el datacenter. Se seleccionó este servidor mediante un acuerdo con los responsables del área, para fines de evaluación de este proyecto. Este activo de la organización fue configurado a modo de prueba con un servidor web Apache versión 2.4.46, que contaba con una formulario web en PHP versión 7.2.21 y una base de datos MySQL versión 8.0.17.

|  |
| --- |
|  |
| Figura 5.8: Topología de despliegue descentralizada RF1 y RF3 |

El ataque consistió en un reconocimiento que partió desde el host ATACANTE ubicado dentro de la Dependencia 1 hacia la víctima llamada Servidor 1 que se encontraba dentro del datacenter.

Para realizar el reconocimiento se utilizaron rutinas de NMAP \cite{nmap} (versión 7.80) configuradas a tal fin: \textit{nmap -T4 -A -v IP\\_VICTIMA} . Los parámetros utilizados fueron:

\begin{itemize}

\item -T4: para un escaneo intensivo (disminuye el tiempo de ejecución entre scripts)

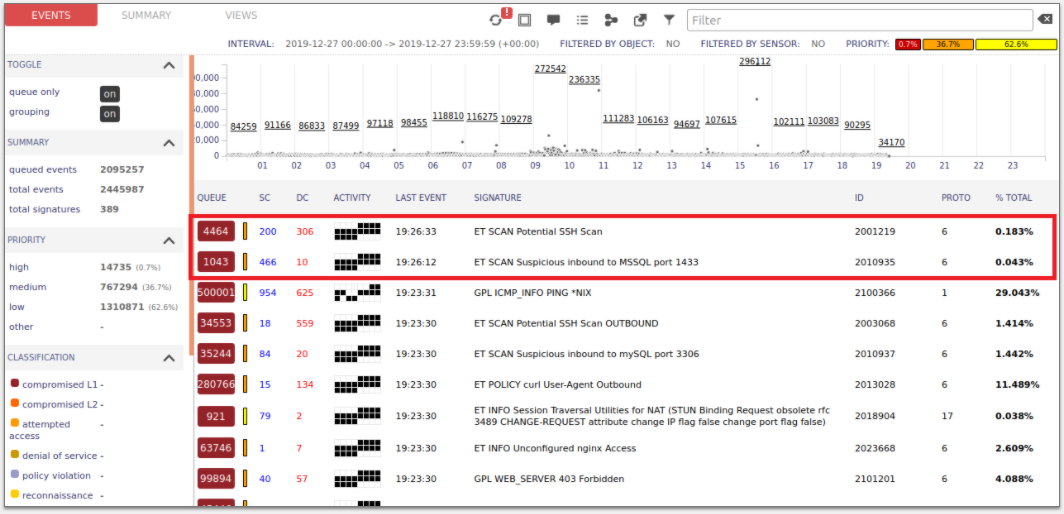
\item -A: habilita la detección del sistema operativo de la víctima y su versión, los scripts de escaneo y traceroute.

\item -v: habilita el modo “verboso”.

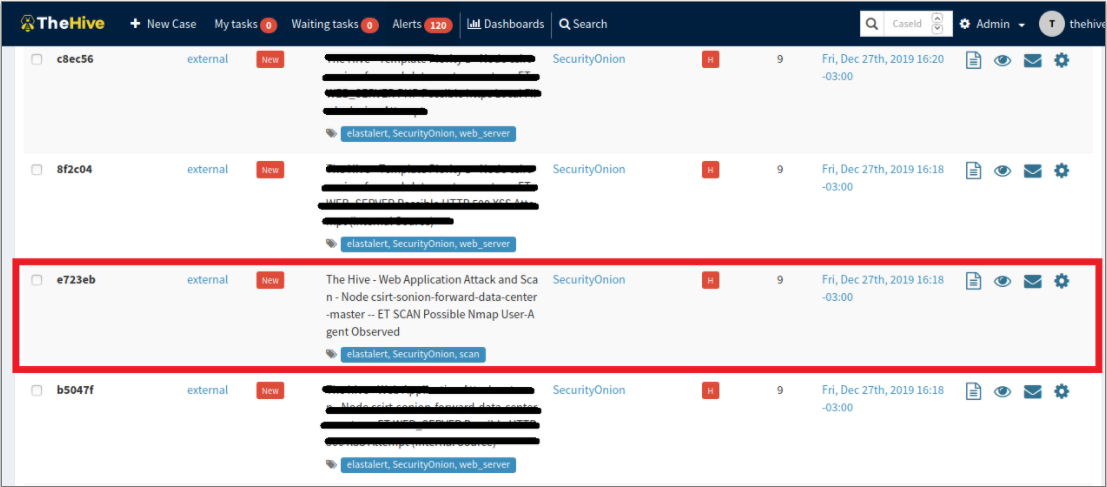
\item IP\\_VICTIMA: es la dirección IP objetivo de este reconocimiento.

\end{itemize}

En el recuadro rojo se puede observar la detección y visualización en Squert del reconocimiento realizado. El resto de la información que se aprecia en la Figura \ref{fig:squert-nmap} corresponde a otros eventos que el sistema estaba detectando. Squert agrupa los eventos según la categoría a la que pertenecen y los clasifica mediante un indicador de colores (barra vertical a la derecha del contador de eventos de cada fila) según su prioridad de atención. Por defecto, Squert tiene su propia categoría de prioridades. Luego se observa la cantidad de eventos según su dirección, hora del último incidente ocurrido junto con su ID, una descripción de la firma del evento y el porcentaje de ocurrencia respecto al total de incidentes detectados.



En la Figura \ref{fig:thehive-nmap} se observa la visualización de la detección en TheHive. Si bien pueden ser similares, Squert se limita a mostrar los ataques similares agrupados en la misma categoría de incidente al que pertenecen, mientras que TheHive recibe las alertas de ElastAlert y presenta los eventos de manera individual. Esto permite a los analistas crear casos con cada incidente, correlacionarse con otros ataques, etc, lo que conduce a una gestión integral de eventos de seguridad.



Se verificó que las direcciones IP reportadas en el incidente por Squert y TheHive, coincidieran con las de la víctima y el atacante. Además mientras se estaba ejecutando el ataque, se hizo un seguimiento mediante Wireshark \cite{wireshark} filtrando las direcciones IP de origen y destino. Esto permitió corroborar el flujo de datos entre ambos.\par

Con esta prueba se verifica el cumplimiento de los requerimientos funcionales 1 y 3: “recolectar y almacenar datos de incidentes de seguridad en la infraestructura de la red corporativa ” y “visualizar las alertas en un tablero de mando”.

## Corolario

En cuanto a la topología de la organización, podemos concluir que, en primer lugar, al tratarse de una topología tipo estrella, la mejor opción para monitorear el tráfico entre las dependencias de la organización y el mundo exterior fue conectar los sensores al Switch 1 (el switch troncal), de manera de poder reenviar el tráfico entre las dependencias y el mencionado switch. En segundo lugar, con el objetivo de ofrecer una defensa de punto final sobre activos de información específicos, fue necesario conectar sensores al switch al que se encuentran conectados estos activos. En este caso, se conectaron sensores al Switch 2.

Se pudo observar que la topología Monolítica fue ineficiente para el uso en entornos de producción, dado que la pila Elastic y los componentes de los sensores IDS demandaron un uso intensivo de hardware. Si bien este tipo de topología fue útil a fines de probar el sistema o para monitorear enlaces de reducido tráfico y poco ancho de banda, resultó totalmente ineficiente en entornos más complejos.

En la organización donde desplegamos nuestro proyecto, debieron ser monitoreados múltiples enlaces y cada uno de estos con un ancho de banda mucho más grande que el conveniente para un nodo monolítico, este tipo de topología demostró ser ineficiente. Las razones se debieron a los enormes requisitos de hardware necesarios para monitorear un enlace de 1 Gbps y considerando que el despliegue final requirió el monitoreo de múltiples enlaces con este ancho de banda, este tipo de topología fue incapaz de escalar utilizando el hardware disponible.

Como consecuencia de la incapacidad de escalamiento horizontal descrita y siendo esta un requerimiento no funcional de este proyecto, se procedió a evaluar el despliegue de una topología distribuida de Security Onion. Los resultados de esta prueba fueron positivos, por lo cual se dio cumplimiento al RNF3, en simultáneo con la verificación de los RF1 y RF3.

# Iteración II: “Información contextual y envió de alertas de seguridad”

Partiendo de la topología definida en la Figura (algo) de Iteración I, se procedió a verificar los requerimientos de prioridad media. Para ello se siguió con la misma metodología de prueba, que consistió en un escaneo de puertos desde una terminal dentro de la Dependencia 1 a un activo ubicado dentro del datacenter llamado Servidor 1.

## Verificación de RF2: Recolección de información contextual

Con el objetivo de recibir y procesar información contextual de los activos que se ven afectados durante un incidente, fue necesario configurar el servidor Master de Security Onion, con otros servicios que permitieran recibir información diferente de la provista por los sensores. Esta información refleja el estado de los servicios en un servidor, su conexión a la red, parámetros de uso de hardware como el nivel de ocupación de disco, temperatura, uso de la memoria RAM, los logs referidos al procesamiento de las peticiones que recibe un servidor, etc. \par

Una anomalía en los datos de esta información no es suficiente por sí misma para confirmar un ataque, pero es útil para indicar que algo puede estar ocurriendo. En el caso de un ataque confirmado, estos datos representan evidencia forense y sirven para enriquecer modelos de amenazas complejas. \par

Se decidió enviar la información contextual disponible mediante Filebeat. Este es un servicio que permite enviar información a Logstash desde múltiples directorios. El proceso comienza cuando el servicio lee línea por línea los archivos de entrada y envía los logs de la misma manera a Logstash. Este los recibe en su puerto 5044 por defecto, posteriormente puede filtrarlos para separar los campos de los logs o bien almacenarlos sin filtrar. \par

Al principio se envió los logs del estado de las aplicaciones de un servidor disponible en la organización, mediante Filebeat, a Logstash y sin un procesamiento posterior. Esto generó que ElastAlert no pudiera realizar una correlación al no identificar información clave, como la que se encuentra en los campos de puertos y direcciones IP de origen y destino, estampas de tiempo, tipo de peticiones, etc.\par

Posteriormente se utilizó el plugin Grok para filtrar los logs que estaba recibiendo Logstash antes de almacenarlos, de manera que la detección basada en correlaciones fue posible.\par

Como se observa en la Figura 6.1, se realizó el envío de logs del Servidor 1 (que se encuentra conectado al Switch 3) mediante Filebeat al nodo Master que se encuentra conectado al Switch 4. Estos logs contenían información de un servidor web Apache, incluyendo consultas realizadas, métodos y otros metadatos de interés. Esta información resultó útil para identificar un ataque de fuerza bruta de estilo “inyección SQL”, que fue simulado utilizando herramientas como “sqlmap”, desarrolladas en Python 3.5. Este tipo de ataque consiste en introducir, de manera iterativa, comandos SQL en un formulario web con el objetivo de lograr recabar información de las bases de datos del Servidor 1 (víctima), modificar e incluso eliminar dicha base de datos. El comando utilizado en sqlmap fue:

*python sqlmap.py -u "http://IP\_VICTIMA/login/index.php?id=101" --batch* donde los flags significan:

* -u: parámetro para especificar una URL
* “[http://IP\_victima/path/al/formulario?etc](http://ip_victima/path/al/formulario?etc)” es el campo que contiene la dirección de un formulario en la página web víctima.
* --batch: es un switch que sirve para elegir la opción por defecto en paginas donde hay multiples parametros. La consecuencia de elegir la opción por defecto es que ahorra tiempo al evitar una elección del usuario.

|  |
| --- |
|  |
|  |

Como se mencionó anteriormente, en primer lugar los logs se almacenaron sin filtrar, lo que ocasionó que no fuese posible realizar algún tipo de correlación con los datos obtenidos. En la Figura 6.2 se puede observar el almacenamiento de estos registros, con esto queda demostrado el cumplimiento del requerimiento funcional 2.

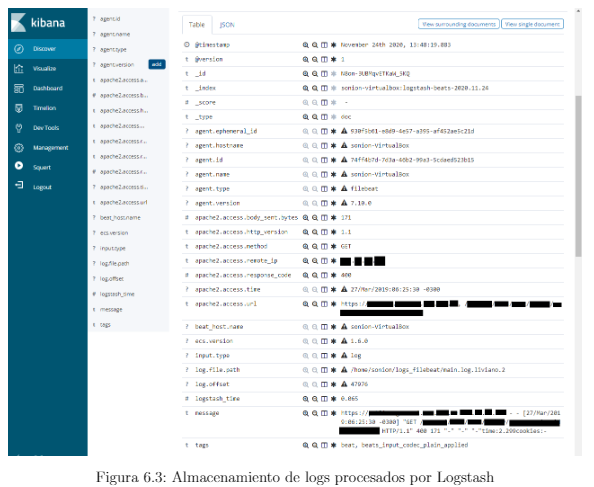


Posteriormente se repitió la experiencia pero esta vez los logs que se recibían en Logstash eran procesados utilizando Grok. Este es un plugin de Logstash que permite filtrar datos sin ningún tipo de estructura y generar información estructurada capaz de ser consultada.

En la Figura 6.x se pudo comprobar que el ataque de inyección SQL fue detectado en Squert.

|  |
| --- |
|  |
| Figura 6.x |

Los resultados en Kibana se ven en la Figura 6.3, donde se registraron las direcciones IP del Servidor 1 (víctima) y la del atacante. \par



Logstash posee varios filtros para todos los tipos de entradas que soporta.

Fue posible editar y administrar estos filtros utilizando archivos de configuración ubicados en la carpeta \textit{/etc/logstash/conf.d.available/}. Para este proyecto se eligió editar el filtro que procesa los datos provenientes de Filebeat, como se indicó anteriormente.\par

Es importante mencionar que la información almacenada en Elasticsearch, es consultada constantemente por ElastAlert. Este último componente es el encargado de correlacionar los campos de los logs y enviar notificaciones.

## Verificación de RF6 y RF4: implementación de un sistema de correlación y envió de alertas de seguridad

El envío de alertas de seguridad y la notificación a los responsables de los activos que se ven afectados, fue realizado en este trabajo mediante ElastAlert. Este componente fue seleccionado ya que permite correlacionar la información presente en los campos de logs guardados en elasticsearch, así como enviar notificaciones cuando se produce la detección de incidentes. \par

ElastAlert es un framework que detecta patrones en los datos consultados a Elasticsearch, como anomalías, picos, etc. Como se mencionó en la sección 4.5, basa su funcionamiento en dos componentes: reglas y alertas. \par

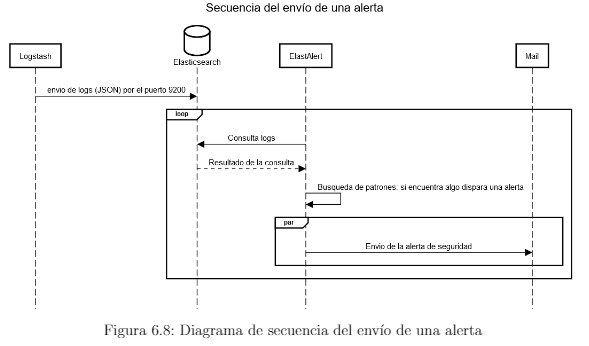
Las alertas consisten en mensajes que permiten notificar a un usuario final o a otro sistema, sobre un evento de seguridad que ha ocurrido. Para esta prueba se decidió realizar nuevamente un escaneo de puertos desde el Atacante ubicado en la Dependencia 1 al Sevidor 1 víctima ubicado en el datacenter. Para esto se configuro a ElastAlert para que notifique por correo electrónico si este detectaba algún tipo de ataque de categoría “scan”.\par

(LO QUE IBA ACA ESTA EN EL ANEXO DESDE EX PAG 78 A 81)

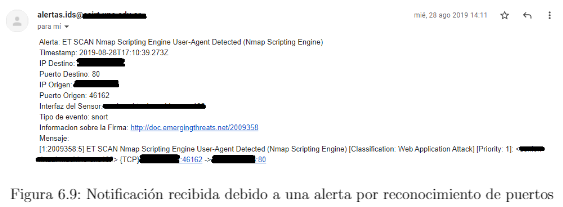
En la Figura 6.8 se observa un diagrama de secuencia del envío de una alerta.

La secuencia comienza cuando los logs que llegan a Logstash son filtrados y enviados de forma estructurada (JSON) al puerto 9200 de Elastisearch. ElastAlert, por otro lado, realiza consultas periódicas a Elasticsearch. Cuando ElastAlert recibe una respuesta a su petición, procede a analizar los resultados en búsqueda de patrones que se puedan identificar en los logs recibidos. \par

Independientemente de la coincidencia o no de algún patrón y la eventual notificación de alerta a los correspondientes responsables, ElastAlert vuelve a consultar a Elasticsearch por otros logs y procede a repetir el procedimiento descrito, mientras este activo como servicio.\par



Los patrones a buscar están definidos en las reglas habilitadas de ElastAlert y si se produce una coincidencia, se procede a enviar una notificación por los medios especificados en las reglas. Como se mencionó anteriormente, para esta prueba el medio elegido fue el correo electrónico de la organización. En la Figura 6.9 se observa el correo electrónico recibido cuando se disparó una alerta sobre un ataque de reconocimiento. Con esto damos por verificado los RF4 y RF6, dado que para que la notificación haya sido enviada, previamente se tuvo que ejecutar una correlación, como se describió anteriormente.



## Capítulo 7: Iteración III: “Priorización de alertas”

El objetivo de asignar distintos niveles de prioridad a las alertas generadas por los eventos que suceden, radica en la naturaleza de estos últimos, su importancia y la gestión de la atención de los analistas del CSIRT. Esto se debe a las necesidades de optimizar el uso de los recursos técnicos y humanos del centro de respuesta a incidentes para cumplir de la manera más eficiente posible con los objetivos y políticas de la organización a la cual pertenece. \par

De esta manera, la naturaleza de los incidentes determina su elegibilidad para una respuesta automatizada al tener en cuenta por un lado su estructura bien conocida y por el otro su alta tasa de repetición en un periodo determinado. En estos casos, sería inutil destinar valiosos recursos tales como la atención de un analista que ya conoce perfectamente la estructura de este tipo de eventos y por lo tanto la respuesta apropiada a él. También se incluyen aquellos casos en los que aún conocida su estructura, el incidente proviene en simultáneo de múltiples fuentes en muy poco tiempo, de manera que la capacidad humana de responder de a un evento a la vez resulta sobrepasada y por lo tanto, ineficiente. Estos son los casos de ataques de reconocimiento y los de denegación distribuida de servicio (DDoS), entre otros. \par

La solución a esta situación se encontró, en primer lugar, en otorgar una baja prioridad a las alertas que estos eventos generaban. Luego se adaptó el sistema de notificaciones para evitar que estas hubiesen abrumado a los analistas y responsables de los activos de información. Finalmente, se consideró la automatización de respuestas a estos incidentes en los casos viables. \par

NUEVA SECCIÓN “Reglas y prioridades”

En Security Onion y otros sistemas, el elemento descriptor que identifica y procesa a cada definición de incidente es la regla. Las reglas comprenden una serie de campos que describen con precisión la naturaleza de un incidente dado y por lo tanto, existen tantas reglas como amenazas detectadas en circulación. \par

Cuando un nuevo malware es descubierto por el equipo de algún CSIRT con la capacidad de investigación suficiente o reportado a un laboratorio apropiado para este fin, es posible realizar un estudio de sus características y una vez identificadas estas últimas, proceder a crear una regla y agregarla al repositorio correspondiente para que otros CSIRT actualicen sus IDS con esta nueva definición y así contar con un filtro (la regla) que permita detectar este malware. Las reglas tienen un conjunto de campos donde se detallan características del paquete y su contexto, tales como el puerto de origen y destino, protocolo empleado, dirección IP, etc y unos campos dedicados a la naturaleza del incidente (clasificación, mensaje, prioridad, etc). Algunos de estos campos son comunes a todas las reglas y permiten agruparlas para administrar eficientemente las alertas generadas cuando una regla coincide con la descripción de un incidente. Dado que estos campos también se pueden considerar observables, es posible utilizarlos por TheHive para gestionar incidentes y crear casos. \par

\begin{section}{Verificación del requerimiento funcional 5: Definición de criterio para priorizar alertas.}

Como se indicó anteriormente, la estructura de las reglas consisten en dos partes bien definidas: un encabezado (header) que es obligatorio y un conjunto de campos opcionales. Dentro del header encontramos la acción (alerta, notificación, etc), el protocolo (tcp, udp), puertos de origen y destino, el sentido del evento (entrante o bidireccional) y las direcciones IP de origen y destino. \par

La segunda parte de las reglas incluye dos tipos de campos: los que describen la naturaleza del evento y aquellos que contienen información del paquete de datos. Dentro del primer grupo encontramos aquellos tales como msg (descripción del evento), sid (id de la firma), classtype (clasificación de reglas o alertas), priority (prioridad de la firma y/o alerta), target (especifica de qué lado está el objetivo, es decir puerto de origen y puerto de destino), entre otros. El segundo grupo contiene datos extraídos que provienen desde de la capa de red hasta la de aplicación de la pila OSI. Se pueden mencionar a los campos “GeoIP” (localización geográfica de la IP), “Fragbits” (presencia del bit de fragmentación), “ACK” (presencia del campo ACK en paquete TCP), “itype” (número del tipo de mensaje ICMP), “http.method” (tipo de método HTTP usado), entre otros.

\begin{figure}[H]

\centering

\includegraphics[width=0.7\textwidth]{./iteracion\_3\_imagenes/figura\_41\_estructura\_regla.png}

\caption{Estructura general de una regla}

\label{fig:figura\_41\_estruc\_regla}

\end{figure}

Como los campos están presentes en todas las reglas, es posible hacer uso de algunos de ellos para agrupar reglas que describen amenazas pertenecientes a un mismo grupo o categoría de malware, intentos de intrusión, reconocimiento, escalado de privilegios, etc y por lo tanto son útiles para gestionar los incidentes. \par

Es posible configurar esta gestión a través de un archivo que relaciona campos como categorías de eventos con prioridades de la alerta generada. Este archivo llamado “classification.config” se encuentra bajo el directorio que almacena las reglas descargadas desde diversas fuentes; en particular relaciona los campos “classtype” con “priority”, de manera tal que cualquier regla cuyo campo classtype contenga a los descritos en este archivo, generará una alerta con prioridad definida también en este. De esta manera, es posible administrar un enorme número de reglas agrupadas en un reducido grupo de categorías y modificar el nivel de prioridad que tendrá en el sistema las alertas que generan. \par

De aproximadamente cuarenta y siete (47) categorías de incidentes disponibles por defecto, consideramos para el máximo nivel de prioridad a siete clasificaciones dado su nivel de ocurrencia y nivel de impacto para la organización.

\begin{itemize}

\item Web-application-attack: esta categoría engloba a un conjunto enorme de malware y ataques a nivel de capa de aplicación. Gusanos, ransomware, ataques de reconocimiento entre otras amenazas comparten esta categoría. Sobre el caso particular de los ataques de reconocimiento, se aplicaron filtros para separarlos de los demás ya mencionados.

\item Unsuccessful User: intentos repetidos de ganar acceso en ciertos activos e infraestructura de la organización.

\item Attempted-dos: intentos de ataque de denegación de servicio y su variante distribuida

\item Known client side exploit attempt: intento de ejecución de exploits en el lado del cliente.

\item Exploit Kit Activity Detected: detección de actividad de un kit de exploits

\item A suspicious filename was detected: detección de nombres de archivos sospechosos

\item Network Trojan: detección de un virus troyano de red.

\end{itemize}

Para verificar el cumplimiento del requerimiento funcional 5, se procedió a editar el archivo classification.config que se encuentra en el path /etc/nsm/rules/. En la Figura \ref{fig:squert-L2} se observa la detección de un incidente de reconocimiento en Squert. Este incidente pertenece a la categoría “attempted-recon” y en el archivo classification.config esta categoría tiene un nivel de prioridad 2, tal como se muestra en la figura (barra vertical naranja). Esto se puede observar con más detalle al analizar los campos de la alerta en Kibana, como se muestra en la Figura \ref{fig:Kibana-L2}. Se remarcó el campo “priority”, donde se vio que efectivamente el valor de la prioridad es 2.

\begin{figure}[H]

\centering

\includegraphics[width=1\textwidth]{./iteracion\_3\_imagenes/squert\_ataque\_vnc\_L2-EDITADO.png}

\caption{Incidente de reconocimiento en Squert. Prioridad nivel 2}

\label{fig:squert-L2}

\end{figure}

\begin{figure}[H]

\centering

\includegraphics[width=1\textwidth]{./iteracion\_3\_imagenes/kibana\_ataques\_L2\_1EDITADO.png}

\caption{Incidente de reconocimiento en Kibana. Prioridad nivel 2}

\label{fig:Kibana-L2}

\end{figure}

\FloatBarrier

Se modificó el archivo \textbf{classification.config} para elevar el nivel de prioridad de los eventos asociados a la categoría “\textit{attempted-recon}”, que pasó del nivel 2 al nivel 1. Posteriormente se reiniciaron los sensores mediante el comando “\textit{so-sensor-restart}” y se procedió a comprobar los resultados de la modificación. Se repitió el ataque de reconocimiento y se pudo observar en las Figuras \ref{fig:squert-L1} y \ref{fig:kibana-L1} que Squert detectaba el ataque con prioridad 1 (barra vertical roja) y en Kibana se observó que el campo “priority” contenía el valor 1. Con esto se da por cumplido el requerimiento funcional 5.

\begin{figure}[H]

\centering

\includegraphics[width=1\textwidth]{./iteracion\_3\_imagenes/squert\_ataque\_vnc\_L1-EDITADO.png}

\caption{Incidente de reconocimiento en Squert. Nivel 1}

\label{fig:squert-L1}

\end{figure}

\begin{figure}[H]

\centering

\includegraphics[width=1\textwidth]{./iteracion\_3\_imagenes/kibana\_ataques\_L2\_2-EDITADO.png}

\caption{Incidente de reconocimiento en Kibana. Nivel 1}

\label{fig:kibana-L1}

\end{figure}

\end{section}

\pagebreak

\begin{section}{Requerimiento no funcional 4: Se requiere un despliegue automatizado de la solución}

Para el despliegue automatizado de la solución se utilizó Ansible 2.8.4. Ansible es una herramienta de automatización de administración de servidores, utilizada en la organización donde se desplegó este proyecto. Esta herramienta permite ahorrar tiempo y esfuerzo al automatizar tareas repetitivas, disminuir errores y aumentar la productividad general cuando se trata del despliegue e instalación de infraestructura básica, como instalar sistemas operativos, realizar la configuración inicial de estos, etc. Es por esta razón que la organización requirió el desarrollo de scripts de instalación de nuestra solución en Ansible.\par

En primer lugar, se desplegó una máquina virtual para probar el resultado de estos scripts. En esta máquina se instaló el sistema operativo Ubuntu Server 16.04 junto a los prerrequisitos para desplegar Ansible: Python en sus versiones 2.7 y 3.5, junto a su módulo pexpect (versión 4.7) y se configuraron las credenciales SSH entre esta máquina y el servidor desde el cual ordenamos el despliegue.\par

Posteriormente se desarrolló un archivo de configuración de Security Onion para su despliegue. Si bien este archivo incluyó numerosos campos y variables, resaltan los correspondientes a la configuración de red, el tipo de rol (Master o Forward) y en base a esto las interfaces de monitoreo o administración, así como configuraciones de almacenamiento de logs, etc.\par

Establecida la comunicación entre la máquina de destino y el servidor, se dio paso al inicio de la ejecución del script de Ansible. Para esto se ejecutó el comando “\textit{ansible-playbook -i hosts -l master so\\_setup.yml --extra-var "target=sonionmaster"\hspace{1 mm} --ask-become-pass}”. En este caso, el comando especifica el despliegue de un nodo Master, sus flags y variables son las siguientes:

\begin{itemize}

\item \textit{ansible-playbook}: almacenan y ejecutan funciones de configuración, despliegue y orquestación de Ansible. Configura los pasos generales de un proceso de infraestructura IT y específica políticas para los sistemas remotos.

\item \textit{- i hosts}: flag que describe el inventario donde se encuentran los hosts. Esta variable \textit{host} es un archivo que describe las direcciones IP de las maquinas disponibles.

\item \textit{- l}: opción utilizada junto al flag “\textit{-i}”, selecciona un subconjunto de hosts del inventario.

\item \textit{--extra-var}: flag que configura variables adicionales como clave/valor o YAML/JSON. En nuestro ejemplo selecciona la opción \textit{sonionmaster} indica que se instaló un nodo Master de Security Onion.

\item --ask-become-pass: flag que solicitó una contraseña de privilegio de usuario cuando fue necesario.

\end{itemize}

En el caso de la instalación de un nodo Forward, hubo que cambiar las opciones del comando que se mencionó anteriormente. El resultado fue: “\textit{ansible-playbook -i hosts -l forward\\_nodes so\\_setup.yml --extra-var "target=sonionforward"\hspace{1 mm} --ask-become-pass}”. \par

En el caso de la instalación de TheHive y Cortex, el proceso es análogo al descrito en el párrafo anterior.\par

El comando para TheHive quedó de la siguiente manera: "\textit{ansible-playbook -i hosts -l thehive setup.yml --extra-var "target=thehiveuser"\hspace{1 mm} --ask-become-pass}", mientras que en el caso de la instalación de Cortex, el comando resultante se muestra a continuación:

"\textit{ansible-playbook -i hosts -l cortex setup.yml --extra-var "target=thehiveuser"\hspace{1 mm} --ask-become-pass}".\par

Entre las acciones que realizó Ansible durante la ejecución de los scripts, se destacaron las de descargar e instalar los paquetes de Security Onion, para luego configurar el sistema operativo utilizando el archivo de configuración mencionado anteriormente. En el caso de la instalación de un nodo Forward, durante la configuración del sistema se debió ingresar la clave ssh para conectarse con el nodo Master.\par

Finalmente se dio por satisfecho este requerimiento no funcional al comprobar que el sistema funcionaba correctamente.\par

\end{section}

## Lista de cosas para hacer 03/12/2020

(minuto 6:13)

Modificar la figura 5.1 lo de centro de análisis por terminales de NOC

(minuto 10:26)

Quizás haya que aclarar en la imagen 5.1 qué enlaces son de 10 G/s y cuales son de 1 G/s

(minuto 11:43 comienza el delirio místico)

(minuto 25:04)

arreglar “baches” imagen 5.3

(minuto 25:43)

corrección del texto

(quede en 28:37)

corrección del texto sección 5.1.1

(minuto 31:25)

Sección 5.1.2 corregir introducción

(minuto 33:46 empieza la nebulosa mal)

(minuto 35:50 creo que leyó cualquier cosa)

(minuto 36:52)

Habla sobre la pérdida de paquetes

(minuto 38:48 fin de la nebulosa)

(minuto 40:54)

Cambiar la intro de la sección 5.2 poner algo como:

“A partir de lo avaluado en el punto anterior, si bien se tomó la decisión de de implementar una arquitectura distribuida para el monitoreo utilizando estas herramientas se comenzó haciendo experiencias en una arquitectura standalone para evaluar el sistema”

Entonces a continuación voy a describir la experiencia de haber implementado o haber desplegado una topología monolítica.

(minuto 42:01)

Con lo anterior en mente no repetir ideas en la sección 5.2.1.

(minuto 44:01)

Agregar que la figura 5.4 corresponde a una prueba de laboratorio

(minuto 45:09)

Cerrar la sección 5.2.1. con lo que tiene marcado

(minuto 47:23)

A la figura 5.4 cambiarle los nombres a las terminales. Entonces

Terminal que envía paquetes ==> Terminal del atacante

Terminal para visualizar datos ==> Terminal del analista

Terminal con Filebeat ==> Generador de logs de eventos

Explicar con flechas el sentido de la comunicación de los logs

(minuto 52:33)

Explicar mejor la parte de la sección 5.2.1 que dice “Una terminal simula ser la de un analista consultando y visualizando datos en Security Onion, mientras que la terminal restante envía logs mediante Filebeat \cite{filebeat}”

(minuto 55:02)

Sección 5.2.1. Elaborar una mini conclusión del despliegue monolítico. Se baja lo que pusimos al principio min(55:16). Cambiar la palabra capacidad

------------------------------------------------------------------------------------------------------------

( minuto 57:17)

Dar detalles del nodo Forward. Describan la experiencia del experimento de la topología distribuida (laboratorio). Agregar imagen con TheHive

(minuto 59:55)

Aclarar en el párrafo donde hablamos de TheHive, qué función cumple Cortex. Hacer una oración como la que hicimos para TheHive.

(minuto 1:01:53) Después de que hice una experiencia con la topología distribuida (sin thehive y cortex) en el laboratorio, voy a llevarla a producción. Digo que sume a TheHive y Cortex en producción y que eso no afecta el rendimiento (porque lo dice el fabricante). No es necesario desarrollar de nuevo el experimento. Ahora cambió de idea de nuevo, quiere que pongamos el experimento con thehive y cortex.

(minuto 1:05:19)

Figura 5.9. Ponerle un circulo al servidor víctima, hacer mas gruesos los enlaces de las dependencias. (1:11:10) agregar prueba exterior (con un router y una casa).

(minuto 1:12:58)

Debajo de la figura 5.8 cambiar “el ataque consistió en un reconocimiento” por “la acción ofensiva consistió en un ataque de reconocimiento”. Cada vez que nos referimos al “ataque” o al “reconocimiento”, debe decir “ataque de reconocimiento”.

(minuto 1:14:46)

Reemplazar la imagen de TheHive por la que tenemos editada (la de las franjas negras en los otros casos).

(minuto 1:15:12)

Figura 5.10 ponerle un recuadro al ataque que queremos mostrar.

(minuto 1:15:38)

Revisar el texto del Corolario, ser más puntual.

(minuto 1:20:32)

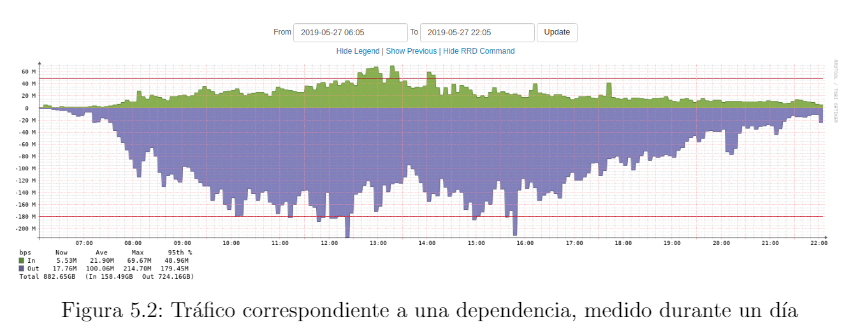
El capítulo 5 lo que estuvo medio flojo fue el proceso de validación que se hizo con un único ataque. (puede ser motivo de la pandemia)

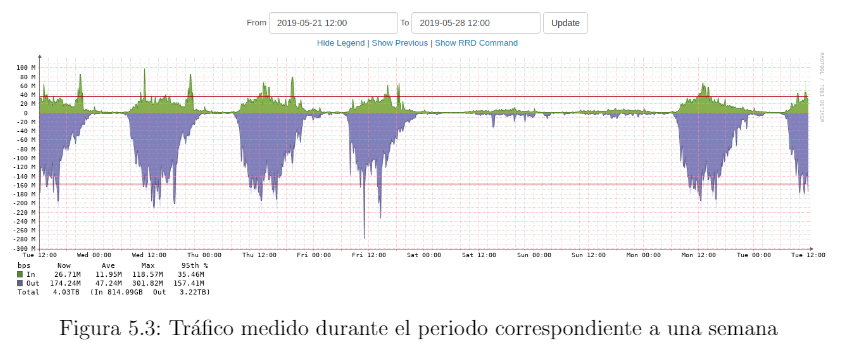
### Definición y configuración de las redes a observar

Se analizó el ancho de banda de las dependencias existentes y se decidió monitorear dos de ellas. Se seleccionaron las de mayor volumen de tráfico en función de mediciones realizadas a lo largo de una semana y de datos históricos. Las dependencias seleccionadas tenían un enlace con ancho de banda de 1 Gbps cada una. \par

Las Figuras \ref{fig:figura\_35\_trafico\_dia} muestra el volumen de tráfico medido en un periodo de tiempo de un día (exceptuando las horas en las que la actividad era mínima), mientras que en la Figura \ref{fig:figura\_36\_trafico\_semana} muestra un periodo de una semana. Estos gráficos fueron obtenidos LibreNMS \cite{librenms} (versión 1.48).

Los datos de las Figuras 5.2 y 5.3 fueron tomados desde la interfaz Gig0 del Switch 1 de la Figura 5.1, esto quiere decir que el volumen de tráfico corresponde a la Dependencia 1. Las interfaces Gig3, Gig-M y Gig-J del Switch 1 tenían un comportamiento similar.\par





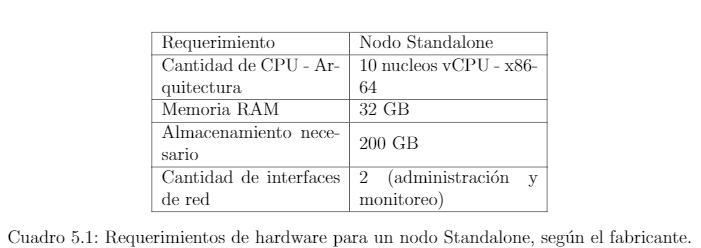
En la Figura 5.3 se observa el tráfico de la interfaz Gig0 en el periodo de una semana, donde la velocidad promedio correspondiente a la entrada es de 11,95 Mbps, mientras que de salida es de 47,24 Mbps. Se observaron picos de 300 Mbps de tráfico.

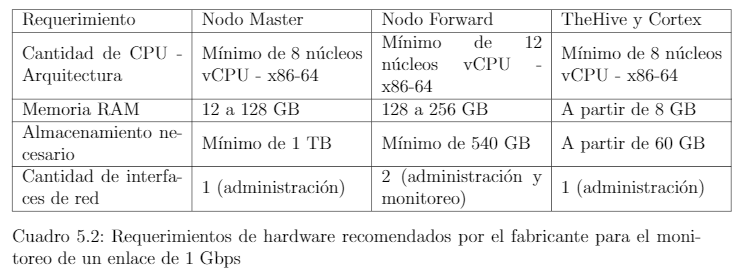
Como la medición corresponde a la interfaz Gig0, el tráfico entrante a esta corresponde al de subida de la Dependencia 1, mientras que el tráfico saliente corresponde al de bajada de esta dependencia.

### 5.1.2 Selección de hardware

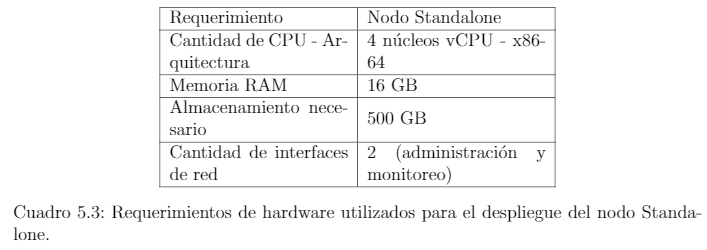
En primer lugar se examinaron los requisitos de hardware mínimos y recomendados por cada uno de los fabricantes. Para esto se tuvo en cuenta el ancho de banda y el tráfico de los enlaces a monitorear, descritos en la sección precedente. Los requerimientos se presentan a continuación, en los Cuadros 5.1 y 5.2 (para nodos Monolítico y Distribuido, respectivamente).

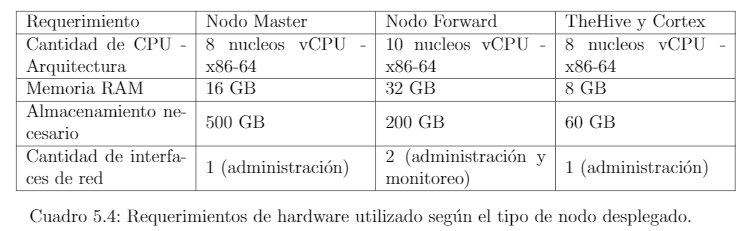
En base a la Figura 5.1, los datos referidos al tráfico y el ancho de banda de los enlaces a monitorear, se dimensionó el hardware necesario para el servidor central, teniendo en cuenta los requerimientos del fabricante y las limitaciones disponibles en la organización.





Debido a las restricciones en la disponibilidad del hardware, se realizó una implementación aproximada, teniendo en cuenta los recursos disponibles en la organización. A continuación se muestran los recursos de hardware utilizados, según se trate del despliegue de una topología Monolítica (Cuadro 5.3) o de una Distribuida (Cuadro 5.4). \par





#### Características del hardware empleado

Se dispuso de varios servidores centrales en los cuales se albergaron las máquinas virtuales de este proyecto. Si bien cada uno de ellos contaba con distinta capacidad de hardware, estas diferencias eran sólo cuantitativas, ya que las características del hardware para todos los servidores eran las mismas.

. A continuación, se detallan las más relevantes:

* En cuanto a los núcleos de CPU virtuales, su frecuencia era de 2.4 Ghz, basados en procesadores físicos Intel Xeon E5620.
* Las memorias RAM pertenecían a la tecnología DDR3 y su frecuencia de refresco era de 2400 MHz.
* Los discos utilizados fueron del tipo mecánico con una velocidad de transferencia para lectura de 196 MB/s y para escritura de 154 MB/s, con interfaz SATA III y 7200 rpm de velocidad de rotación.

### Pérdidas de paquetes

Como consecuencia de no satisfacer los requerimientos de hardware recomendados por el fabricante, para los nodos que contienen sistemas IDS, se detectó una pérdida de paquetes que varía según el volumen de tráfico. El estudio de este problema fue realizado por otro de los grupos, que llevó a cabo su Proyecto Integrador sobre sistemas IDS. Se realizaron pruebas de velocidad de tráfico y el porcentaje de paquetes que se perdían a medida que variaba la velocidad, como se ve en el Cuadro 5.x.

|  |  |  |
| --- | --- | --- |
| Velocidad (Mbps) | Pérdidas de paquetes (%) | Alertas no detectadas (%) |
| 800 | 25 | 40 |
| 400 | 12 | 20 |
| 200 | 0 | 0 |
| Cuadro 5.x | | |

La pérdida de paquetes por parte de los IDS tiene como consecuencia la no detección de alertas. En base a pruebas con duración de un día cada una, se estimó el porcentaje de alertas no detectadas según el porcentaje de pérdidas de paquetes. Los resultados de estas pruebas se observan en el Cuadro 5.x

## Monitoreo de red

A pesar de conocer, por la sección anterior, que el rendimiento de una topología monolítica no es óptimo en entornos de alto rendimiento, se decidió desplegar esta topología en primer lugar para familiarizarnos con la solución.

### Topología Monolítica de Monitoreo

En la Figura \ref{fig:figura\_33\_a}, se observa una topología de prueba desarrollada en una red aislada, a modo de experiencia de laboratorio, en el cual se desplegó la solución con una arquitectura monolítica. \par

El experimento consistió en demostrar la capacidad de monitoreo y detección de incidentes de Security Onion, mediante el despliegue de un nodo Monolítico y tres terminales de computadoras, una de las cuales (C) simuló ser el atacante.\par

En primer lugar, se conectaron las terminales “A” y “B” (la del analista y el generador de logs, respectivamente) a un switch al cual se conectó, mediante su interfaz “Eth1”, uno de los enlaces del servidor de Security Onion. Este enlace, llamado “Eth0” en la Figura \ref{fig:figura\_33\_a}, sirvió a fines de que el analista en el terminal “A” simule consultas y visualice datos en Security Onion, mientras que la terminal “B” envíaba logs mediante Filebeat \cite{filebeat}, conteniendo información contextual de los activos de la red. \par

%COLOCAR FIGURA ACA

El tercer terminal (C) se conectó mediante un enlace a la interfaz “Eth1” del nodo Monolítico, enviando tráfico simulado mediante PCAPS conteniendo acciones ofensivas. Para lograr esta acción, en el terminal C se utilizó TCPreplay \ref{tcpreplay} para reenviar los PCAPS mencionados. Esta interfaz estaba configurada para monitorear el tráfico en busca de paquetes que pudieran contener una acción ofensiva.

Posteriormente se colocó en producción un nodo Monolítico. En la Figura \ref{fig:iter1\_top\_m\_unc} se observa la topología de la organización incluyendo la conexión con este nodo. Este tiene su interfaz de monitoreo conectada al Switch 3, donde recibe el tráfico reenviado que se produce entre la Dependencia 1 y el mencionado switch.

%Colocar ACA FIGURA TOPO UNC NODO MONOLITICO

Se realizaron pruebas de los requerimientos funcionales 1 y 3: se comprobó el funcionamiento de la recolección y almacenamiento de datos de incidentes de seguridad, al recibir y analizar tráfico proveniente de la Dependencia 1. \par

### 

### Modificaciones a la sección Topología Distribuida de Monitoreo

Luego de la incorporación de TheHive, se volvió a repetir el experimento detallado anteriormente, esta vez con el objetivo de que TheHive recibiera las alertas generadas en el nodo Master mediante ElastAlert. A diferencia del experimento anterior, en el que el nodo Master recibía logs filtrados desde Filebeat, en esta ocasión se lanzó al nodo Forward un ataque simulado por su interfaz de monitoreo.\par

De esta manera, cuando el nodo Forward identificó el ataque, notificó al nodo Master del mismo. En el nodo Master, ElastAlert generó alertas que fueron enviadas al servidor de TheHive, quien las presentó a los analistas junto a información contextual que intentó correlacionar con información presente en su base de datos.\par

%Aca viene la imagen con la prueba de thehive

\begin{figure}[H]

\centering \includegraphics[width=1\textwidth]{./iteracion\_1\_imagenes/figura\_33\_c\_topologia\_de\_prueba\_3.png}

\caption{Experiencia de laboratorio de la topología distribuida con todos sus componentes}

\label{fig:topologia\_distribuida\_2}

\end{figure}

\FloatBarrier

### Modificaciones al Corolario del capítulo 5

En cuanto a la topología de la organización (Figura \ref{fig:iter1\_top\_unc}), podemos concluir que, la mejor opción para monitorear el tráfico entre las dependencias de la organización y el exterior fue conectar los nodos Forward al Switch 1 (el switch troncal), para recibir el tráfico reenviado entre las dependencias y el mencionado switch. Por otro lado, con el objetivo de lograr una defensa de punto final sobre activos de información específicos, fue necesario conectar un nodo Forward al switch (Switch 2) al que se encuentran conectados estos activos. Estas conexiones se vieron reflejadas en la Figura \ref{fig:iter1\_ver\_RF1\_RF2}. \par

Se observó que la topología Monolítica fue ineficiente para el uso en entornos de producción, dado que la pila Elastic y los componentes de los sensores IDS de este nodo demandaron un uso intensivo de hardware. Si bien este tipo de topología fue útil a fines de probar el sistema o para monitorear enlaces de reducido tráfico y poco ancho de banda, resultó ineficaz en entornos más complejos. Esto se debió a los enormes requisitos de hardware necesarios para monitorear un enlace de 1 Gbps y considerando que el despliegue final requirió el monitoreo de múltiples enlaces con este ancho de banda, este tipo de topología fue incapaz de escalar utilizando el hardware disponible.\par

Como consecuencia de la incapacidad de escalamiento horizontal descrita y siendo esta un requerimiento no funcional de este proyecto, se desplegó una topología distribuida de Security Onion. Los resultados del monitoreo fueron positivos, por lo cual se dio cumplimiento al RNF3, en simultáneo con la verificación de RF1 y RF3.\par

### Capítulo 6: Verificación de RF2.

Información contextual, ¿que es? ¿para qué sirve?

¿para que la queremos? → correlación entre la información contextual (logs antes de un ataque confirmado) e información contextual (logs después de un ataque confirmado). ¿Quien hace la correlación de estos logs? Elastalert

¿Cómo se puede demostrar esto? Ataque DDOS, captura de kibana logs parseados

La información contextual de un activo de información está constituida por todos aquellos datos que reflejan, en un momento particular, el estado del sistema. Entre los ejemplos se encuentran los logs de las peticiones recibidas, el estado de los servicios habilitados, la conexión a la red, parámetros de uso de hardware tales como el nivel de ocupación de disco, temperatura, uso de la memoria RAM, entre otros. Una anomalía en estos datos es útil para indicar que algo puede estar ocurriendo, tal como una acción ofensiva encubierta en operaciones habilitadas. Por ejemplo, un ataque DDoS implica el aumento exponencial de la frecuencia normal de transacciones. En el caso de un ataque no detectado por un IDS, estos datos sirven para alertar de que una posible acción ofensiva está teniendo lugar. \par

Con el objetivo de recibir esta información, el servidor Master de Security Onion recolectó logs provenientes de servicios de “log shipping” (proceso de automatización del respaldo de archivos de logs de transacciones). Para este proyecto se optó por Filebeat. Este es un servicio que permite enviar información a Logstash desde múltiples directorios. El proceso comienza cuando el servicio lee línea por línea los archivos de entrada en un activo y envía los logs de la misma manera a Logstash (servidor Master). Este los recibe en su puerto 5044 por defecto, posteriormente puede filtrarlos para separar los campos de los logs o bien almacenarlos sin filtrar. \par

La experiencia inició con el envío de los logs del estado de las aplicaciones de un servidor disponible en la organización, mediante Filebeat, a Logstash y sin un procesamiento posterior. Esto generó que ElastAlert no pudiera realizar una correlación al no identificar información clave, como la que se encuentra en los campos de puertos y direcciones IP de origen y destino, estampas de tiempo, tipo de peticiones, etc. Posteriormente se utilizó el plugin Grok \cite{grok} para filtrar los logs que estaba recibiendo Logstash antes de almacenarlos, con el fin de realizar detecciones basadas en correlaciones utilizando ElastAlert.\par

Como se observa en la Figura \ref{fig:figura\_iter2\_ataque}, se realizó el envío de logs del Servidor 1 (que se encuentra conectado al Switch 3) mediante Filebeat al nodo Master que se encuentra conectado al Switch 4. Estos logs de accesos contenían información de un servidor web Apache, incluyendo consultas realizadas, métodos y otros metadatos de interés. Esta información resultó útil para identificar un ataque de aplicación web, de estilo “inyección SQL”, que fue realizado utilizando herramientas como “sqlmap” \cite{sqlmap}, desarrolladas en Python 3.5. Este tipo de ataque consiste en introducir, de manera iterativa, comandos SQL en un formulario web con el objetivo de lograr recabar información de las bases de datos del Servidor 1 (víctima), modificar e incluso eliminar dicha base de datos. El comando utilizado en sqlmap fue:

\textit{python sqlmap.py -u "http://IP\\_VICTIMA/login/index.php?id=101" --batch} donde los flags significan:

\begin{itemize}

\item \textit{-u}: parámetro para especificar una URL.

\item \textit{“http://IP\\_victima/path/al/formulario?etc”} es el campo que contiene la dirección de un formulario en la página web víctima.

\item \textit{--batch}: es un switch que sirve para elegir la opción por defecto en paginas donde hay múltiples parámetros. La consecuencia de elegir la opción por defecto es que ahorra tiempo al evitar una elección del usuario.

\end{itemize}

\begin{figure}[H]

\centering

\includegraphics[width=1\textwidth]{./iteracion\_2\_imagenes/figura\_iter2\_ataque.png}

\caption{Topología de despliegue descentralizada RF2, RF6 y RF4-FILEBEAT.png}

\label{fig:figura\_iter2\_ataque}

\end{figure}

\FloatBarrier

En la Figura \ref{fig:iter2\_logs\_crudos} se puede observar el almacenamiento de registros sin filtrar, con esto queda demostrado el cumplimiento del requerimiento funcional 2.

\begin{figure}[H]

\centering

\includegraphics[width=1\textwidth]{./iteracion\_2\_imagenes/1\_kibana\_logs\_1EDITADA.png}

\caption{Almacenamiento de logs sin procesar}

\label{fig:iter2\_logs\_crudos}

\end{figure}

\FloatBarrier

Posteriormente se repitió la experiencia pero esta vez los logs que se recibían en Logstash eran procesados utilizando Grok. Esto permitió generar información estructurada capaz de ser consultada. \par

En la Figura \ref{fig:figura\_squert-sql} se pudo comprobar que el ataque de inyección SQL fue detectado en Squert.

\begin{figure}[H]

\centering

\includegraphics[width=0.7\textwidth]{./iteracion\_2\_imagenes/squert-sql-injection.png}

\caption{Detección de un ataque de inyección SQL en el panel de Squert}

\label{fig:figura\_squert-sql}

\end{figure}

Los resultados en Kibana se ven en la Figura 6.3, donde se registraron las direcciones IP del Servidor 1 (víctima) y la del atacante. \par

\begin{figure}[H]

\centering

\includegraphics[width=1\textwidth]{./iteracion\_2\_imagenes/kibana\_logs\_parseados\_2EDITADO.png}

\caption{Almacenamiento de logs procesados por Logstash}

\label{fig:iter2\_logs\_filtrados}

\end{figure}

\FloatBarrier

Es importante mencionar que la información almacenada en Elasticsearch, es consultada constantemente por ElastAlert. Este último componente es el encargado de correlacionar los campos de los logs y enviar notificaciones.

\end{section}

\pagebreak

## Verificación de RF6 y RF4: implementación de un sistema de correlación y envió de alertas de seguridad

El envío de alertas de seguridad y la notificación a los responsables de los activos que se ven afectados, fue realizado por ElastAlert. Este permite correlacionar la información presente en los campos de logs guardados en Elasticsearch, así como enviar notificaciones cuando se produce la detección de incidentes. \par

ElastAlert es un framework que detecta patrones en los datos consultados, como anomalías, picos, etc. Como se mencionó en la sección 4.5, basa su funcionamiento en dos componentes: reglas y alertas. \par

Las alertas consisten en mensajes que permiten notificar a un usuario final o a otro sistema, sobre un evento de seguridad. Para esta prueba se realizó otro escaneo de puertos desde el “atacante” ubicado en la Dependencia 1 dirigido al Sevidor 1 (víctima), ubicado en el datacenter. Para esto se configuro a ElastAlert para que notifique por correo electrónico en el caso de detectar algún tipo de ataque de categoría “scan”.\par

En la Figura \ref{fig:iter2\_diagrama\_envio\_alertas} se observa un diagrama de secuencia del envío de una alerta. Esta comienza cuando los logs que llegan a Logstash son filtrados y enviados de forma estructurada (JSON) al puerto 9200 de Elastisearch. ElastAlert, por otro lado, realiza consultas periódicas a la base de datos mencionada. Cuando ElastAlert recibe una respuesta a su petición, procede a buscar patrones que se puedan identificar en los logs recibidos. \par

Independientemente de la coincidencia o no de algún patrón, ElastAlert vuelve a consultar a Elasticsearch por otros logs y procede a repetir el procedimiento descrito, mientras este activo como servicio. En el caso de encontrar un patrón en los datos recibidos, se enviará una notificación a los responsables del activo afectado.\par

\begin{figure}[H]

\centering \includegraphics[width=1\textwidth]{./iteracion\_2\_imagenes/2-diagrama-de-secuencia-envio-alerta.png}

\caption{Diagrama de secuencia del envío de una alerta}

\label{fig:iter2\_diagrama\_envio\_alertas}

\end{figure}

En la Figura \ref{fig:iter2\_notificacion\_alertas} se observa el correo electrónico recibido cuando se disparó una alerta por el ataque de reconocimiento. Con esto se verificaron RF4 y RF6.\par

\begin{figure}[H]

\centering \includegraphics[width=1\textwidth]{./iteracion\_2\_imagenes/notificacion\_alertas\_1EDITADO.png}

\caption{Notificación recibida debido a una alerta por reconocimiento de puertos}

\label{fig:iter2\_notificacion\_alertas}

\end{figure}