## 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.