ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE TELECOMUNICACIÓN

INGENIERÍA EN INFORMÁTICA

Security Sensor

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE TELECOMUNICACIÓN

INGENIERO EN INFORMÁTICA

Security Sensor

- Departamento: Teoría de la señal, Telemática y Comunicaciones
- Director del proyecto: Gabriel Maciá Fernández
- Autor del proyecto: Moisés Gautier Gómez

Granada, 21 de mayo de 2016

Fdo: Moisés Gautier Gómez

Índice general

Índice de tablas

Índice de figuras

Capítulo 1

Introducción

Estar preparado para la guerra
es uno de los medios más eficaces
para conservar la paz
George Washington

En el siglo XXI toda pasa por ser digital y sino, ya es que lo era antes de llegar a este punto. Quizás muchas de las tecnologías que hoy conocemos se basen en un sistema informatizado, ya sea en su formato de software o sistema embebido. Y es que todo pasa por ser una herramienta software diseñadas para un propósito en concreto: un mecanismo de apertura de puertas mediante tarjetas rfid, procedimientos industriales o aplicados a alguna infraestructura crítica o de bien común, una aplicación del tiempo en tú terminal móvil, el propio sistema operativo con el que se puede leer el documento, etc.

Hay un sinfín de aplicaciones software que hacen nuestro día a día más llevadero y más fácil. Pero hay un punto que no todos conocen y es la necesidad de saber cómo se ha creado ese producto o cómo funciona realmente por su interior. En ése interior, a veces, podemos encontrar cosas que no estaban predestinadas a tener ese comportamiento y debido a ése comportamiento anómalo o imprevisto se generan situaciones de incertidumbre en las que el ser humano debe estar capacitao a afrontar. Dichas situaciones se suelen conocer con el término anglosajón de "bugz sobre los bugs hay una especial categoría que se denominan fallas de seguridad o critical/several bugs.

Estas situaciones contractuales provocan que nuestro sistema, sea el que fuese, actúe de forma inesperada ante un input de información permitido o legítimo permitiendo un uso inadecuado de los recursos a los que se acceden mediante la aplicación. De este concepto o problema, surge en gran medida, el término de seguridad informática el cuál intenta abarcar y dar solución a estos problemas que pueden ir desde un simple fallo de desarrollo a un fallo crítico que comprometa la seguridad o confidencialidad de los documentos de una empresa o gobierno.

Debido a esta problemática, surge la necesidad de analizar, monitorizar y generar sistemas de seguridad perimetral que permita a las empresas ver que tipo de tráfico interno

se genera, que tipo de tráfico externo tiene y cómo se hace uso de él (navegación hacia el exterior, tuneles vpn, conexiones remotas a dispositivos, etc). Así pues, se podría decir que para obtener este tipo de eventos sobre protocolos, tráfico, dns, ips, vpns,.. se tienen que configurar dispositivos de seguridad para la recolección de estos tipos de inputs o fuentes.

Una de las fuentes más conocidas dentro del mundo de la informática es el firewall, pero no así de las de un uso más extendido dentro del mundo doméstico sino el comercial o corporativo. Y dentro de los muchos tipos de softwares enfocados a tráfico (firewall) se encuentra el paquete de las distribuciones GNU/Linux iptables (software fruto del proyecto Netfilter para el kernel de GNU/Linux). Con esta herramienta se pueden definir políticas de filtrado de tráfico para cualquier tipo de protocolo tcp/udp que queramos limitar entre el exterior y nuestra máquina y viceversa. Además, estas políticas nos permiten derivar dicho tráfico a archivos que podemos manipular obteniendo así los eventos que representan al tráfico generado por una máquina conectada a una red, que posteriormente podemos manipular para generar estadísticas o tipos de uso para una red.

Aquí nace éste proyecto final de carrera. La solución que se busca desarrollar se denominaría Sensor de Seguridad (Security Sensor). De forma muy general, éste software se encargará de monitorización información de una máquina (logs, red (iptables), etc), almacenarla, visualizarla y además realizar un procesado de la misma con un algoritmo de obtención de características específico.

1.1. Objetivos

El objetivo principal del proyecto es controlar los tipos de conexiones entrantes y salientes de una máquina en sus diferentes protocolos de comunicación y mecanismos de gestión.
La aplicación deberá cumplir los siguientes requisitos:

- Ser una herramienta multiplataforma y que permita a cualquier usuario definir sus propias interfaces de gestión de eventos.
- Dotar de funcionalidad gráfica que permita extraer información en tiempo real con gráficas o mecanismos visuales (en web) del sistema de base de datos que ha procesado los inputs de las fuentes para las que ha sido configurada.
- Dotar de una api interna que nos permita extraer información en tiempo real en un formato uniforme para la web o para que cualquier usuario pueda usar la funcionalidad del proyecto para su propio beneficio usando herramientas generadas en el back-end para otro tipo de aplicaciones.
- Ser parte de un todo, en el que el todo sea un SIEM capaz de obtener información de las diferentes sondas o módulos, que en este caso, sería la solución desarrollada.

- Desarrollar una sonda para procesar eventos logs de firewall.

1.2. Contexto: Seguridad informática

La seguridad informática es el proceso de mantener un aceptable nivel de percepción frente a un riesgo. Así pues ninguna organización se puede considerar "segura" en cualquier momento, más allá de la última comprobación que se realizo dentro de su política de seguridad.
Un proceso seguro se encuadra dentro de las siguientes 4 etapas: Evaluación, Protección, Detección y Respuesta:

- Evaluación: es la preparación para las otras 3 etapas. Se considera cómo una acción separada porque se relaciona con las políticas, procedimientos, leyes, reglamentos, presupuestos y otras funciones de gestión, además de la propia evaluación técnica enfocada a la seguridad. No tener en consideración estos supuestos anteriores, podría dañar las etapas del diseño.

- Protección: es la aplicación de contramedidas para reducir la probabilidad de tener una situación comprometida. La prevención es un término equivalente, pero cabe decir que en la mayoría de casos este concepto suele inducir al error (o al fallo).

- Detección: es el proceso de identificación de intrusos. Una intrusión se puede considerar cómo una violación de una política de seguridad o cómo un incidente de seguridad a nivel de software/dispositivo.

- Respuesta: es el proceso de validar los inputs recogidos por la detección para tomar medidas que solucionen las intrusiones. El primer enfoque que debemos realizar consiste en restaurar la funcionalidad dañada y seguir recopilando información para tener claras las evidencias del atacante sobre nuestro sistema y poder así emprender las acciones legales que correspondan.

1.2.1. ¿Qué es el riesgo?

Cómo ya se mencionó en la definición de seguridad, el riesgo, es la posibilidad de sufrir un daño o pérdida. El riesgo, es una medida de peligro para un activo. Un activo se puede considerar cualquier cosa de valor, que en el contexto en el que nos encontramos se refiere a la información, hardware, propiedad intelectual, prestigio y reputación. Esto se puede definir cómo, "riesgo de comprometer la integridad de la base de datos de un cliente" o "riesgo de una denegación de servicio al portal eléctronico de una entidad pública de gestión". Se suele expresar en términos de una ecuación de riesgo, dónde:

$$Riesgo = Amenaza \times Vulnerabilidad \times Valor\ del\ activo$$

- Amenaza: es el conjunto de las capacidades e intenciones de un atacante para explotar una vulnerabilidad en activo.

- Vulnerabilidad: es una debilidad de un activo que podría conducir a la explotación del mismo. Las vulnerabilidades se introducen en los activos a través de un mal diseño, implementación o de contención de datos.

- Valor del activo: es una medida de tiempo en relación a los recursos necesarios para reemplazar un activo o restaurarlo a su estado anterior funcional.

1.3. Alcance

El proyecto Security Sensor dará cómo resultado una aplicación software con interfaz visual web, que cumplirá con todos los objetivos y especificaciones indicados en el apartado anterior.
La aplicación se distinguirá en varias partes:

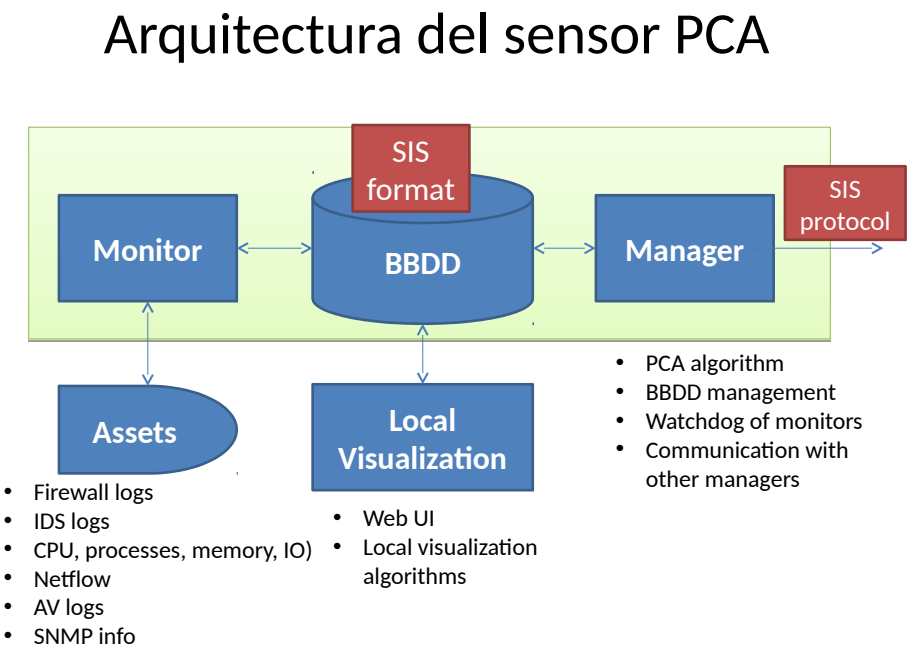
- Configuración y parametrización de los eventos logs de firewall en el dispositivo.

- Recolección de logs mediante rsyslog y su posterior procesamiento.

- Almacenaje en BD para su posterior manipulación interna o externa (api).

- Extracción de características de los eventos y visualización de los mismos mediante un servicio web.

- Aplicación de un algoritmo de procesamiento para comprimir la información en formato módulo para ser servida a un nodo central de gestión (SIEM).



1.4. Visión global

En cuanto a la estructura de esta memoria del proyecto final de carrera, tras éste capítulo dónde se presentan los objetivos y la visión en general del proyecto, se expone el estado del arte y el análisis de requisitos previos al desarrollo software.
En el capítulo siguiente, veremos la etapa del diseño de software así cómo posterior evaluación del mismo.
Finalmente, se presentan las conclusiones generales obtenidas una vez realizado el proyecto, así también la planifiación del mismo y estimación de costes.
Además, se presentan las referencia bibliográficas dónde se incluyen las fuentes consultadas para la elaboración de este proyecto, un resumen que engloba las generalidades fundamentales de la aplicación, una guía de utilización (manual de usuario), una guía de instalación, un compendio del software utilizado para el desarrollo y otro de los lenguajes de programación, y finalmente, la licencia completa del documento.

Capítulo 2

Estado del arte

Actualmente existen diversos tipos de herramientas que realizan tareas de recolección, filtrado y gestión de eventos dentro un sistema. Las que se han podido analizar y comprobar han sido las siguientes:

2.1. Lookwise

Lookwise es una herramienta corporativa que hace las funciones de SIEM en materia de gestión de seguridad, Big Data y cumplimiento normativo (ISO/LOPD).

2.1.1. Características

Gestión centralizada

- Interfaz gráfica de administración y operación centralizada.
- Creación y distribución de políticas de forma remota.
- Integración de alertas y explotación de resultados.
- Cuadro de mandos de seguridad.
- Visibilidad en base a roles y permisos.
- Integración con sistemas SIEM.

Comunicaciones

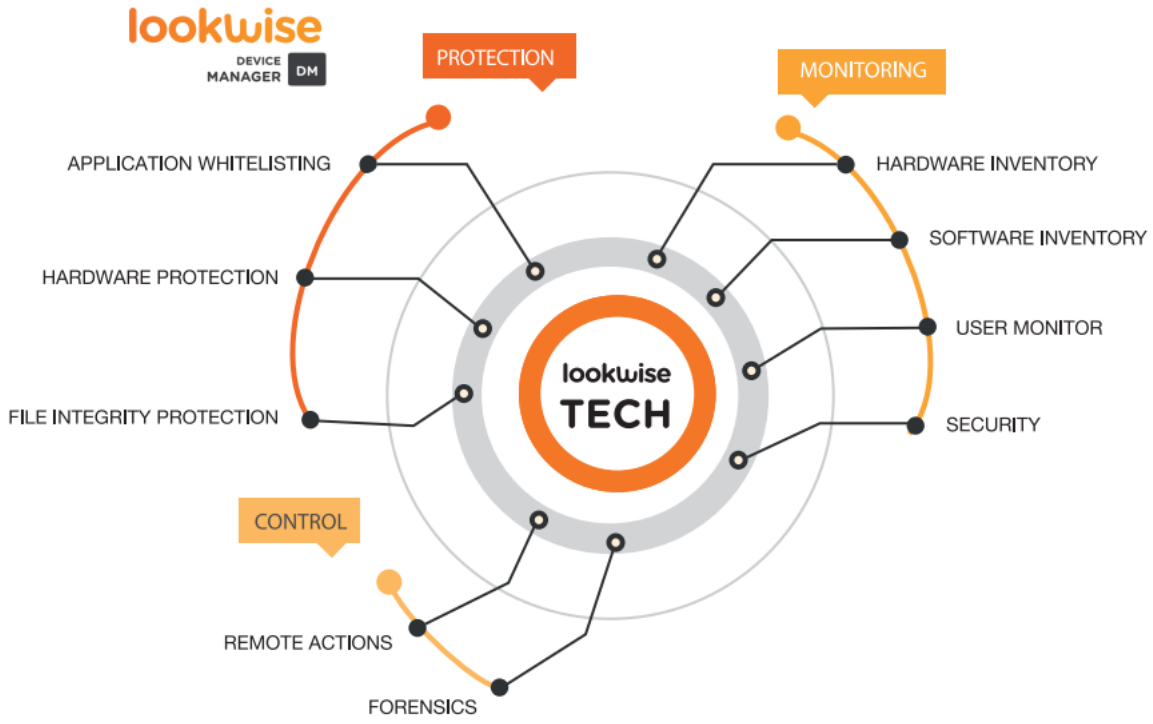
- Autenticadas y cifradas
- Ininterrumpidas
- Comprimidas

Plataformas soportadas

- Familia Windows XP (Windows Kernel 5)
- Familia Windows 7 (Windows Kernel 6)

Arquitectura

- Arquitectura Distribuida.
- Arquitectura Modular.
- Flexible y escalable.
- Balanceo de carga.
- Despliegue remoto de funcionalidades y actualizaciones.

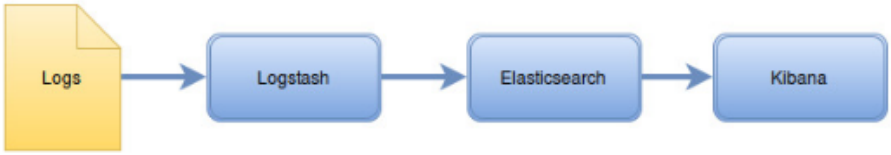


2.1.2. Análisis de la herramienta

La herramienta de análisis de incidencias y vulnerabilidades hace la mismas funcionalidades de un SIEM pero con una capa más enfocada a sistemas PCI y de infraestructuras críticas. Gestiona eventos del sistema y los correla con alertas predefinidas internamente o que se hayan incluido cómo especificación del cliente. Detecta fallos en el Active Directory y nos genera un sistema de informe con las incidencias más graves de cara a la parte de consultoría de incidentes por parte el equipo interno.

2.2. ELK Stack

La pila ELK se basa en una solución open-source de tres productos bien diferenciados que se relacionan entre sí de la siguiente manera:

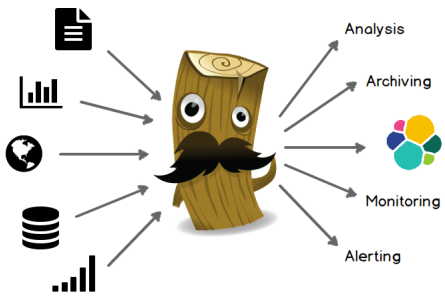


- Elasticsearch para la búsqueda de datos y análisis en profundidad de un sistema.
- Logstash para el registro centralizado de logs y su posterior normalización y enriquecimiento de datos.
- Kibana cómo herramienta de visualización de los datos recolectados y procesados anteriormente según las especificaciones que queramos para el filtrado.

2.2.1. Elasticsearch

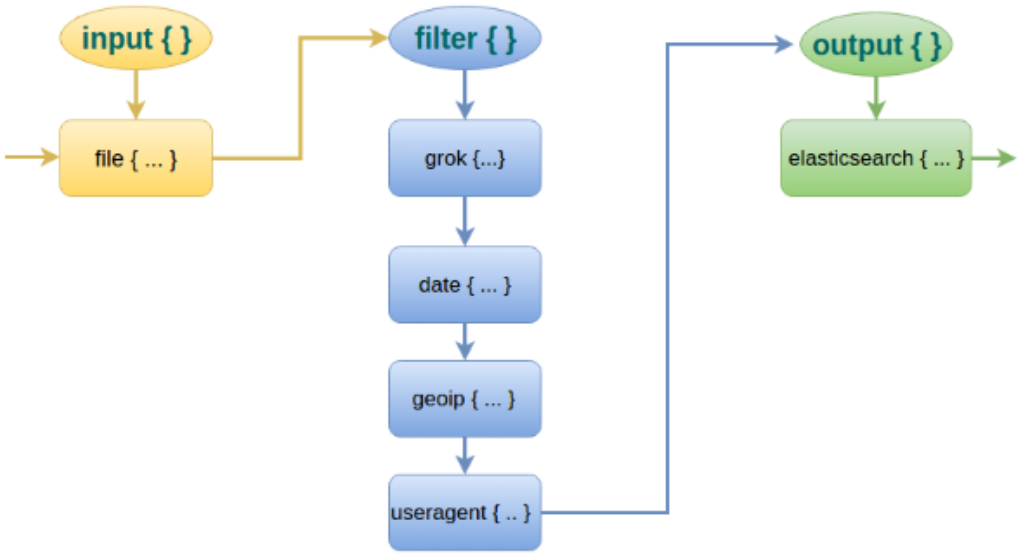
Elasticsearch es un motor open-source de búsqueda y análisis de información de gran escalabilidad. Esta herramienta permite almacenar, buscar y analizar grandes volúmenes de datos de forma rápida y en tiempo real. Se suele utilizar cómo motor/tecnología subyacente de otras aplicaciones (wrapers), permitiendo así realizar funciones de búsqueda complejas de una manera más ágil.

2.2.2. Logstash



Logstash es un motor open-source de recopilación de datos con capacidad de multihebrado en tiempo real. Esta herramienta puede unificar dinámicamente datos de diferentes fuentes y normalizar dichos datos para los outputs de nuestra elección. Además nos permite filtrar y discretizar todo los datos recolectados para obtener información

analítica que pueda ser visualizada.



Con logstash, cualquier tipo de evento puede ser enriquecido y transformado con un amplio repertorio de inputs/filtros y los datos de salida, se pueden modificar dependiendo del tipo de software sobre el que queramos introducir esos datos (plugins y códecs).

2.2.3. Kibana

Kibana es una plataforma open-source de análisis y visualización de datos diseñada para trabajar con Elasticsearch. Kibana se utiliza para para buscar, ver e interactuar con datos almacenados en los índices o base de datos de Elasticsearch. De esta forma se puede realizar fácilmente un análisis avanzado de datos que permita visualizarlo en una gran variedad de gráficas, tablas y mapas.

Kibana hace que sea fácil de entender y procesar grandes volúmenes de datos. Mediante su interfaz basada en un cliente web, nos permite crear de forma simple y ágil filtros sobre los datos extraidos mediante consultas a la bd de Elasticsearch en tiempo real.

2.2.4. Análisis de la herramienta

La pila ELK, cómo bien se ha comentado en los tres puntos anteriores nos permite recolectar, procesar y correlar cualquier tipo de logs que genere nuestro sistema de una manera visual, ágil y fácil de entender.

El motor de indexación de datos, Elasticsearch, nos permite obtener una ejecución en tiempo real de los logs del sistema así cómo poder escalar dicho volumen de datos dependiendo de la situación. El único inconveniente que podría sacar de este fragmento de la pila es que el sistema de referenciación de documentos internos es mediante json. Es un formato muy versátil pero incapaz de tener funcionalidad por si sola.

Después tenemos Logstash que es el encargado de hacer de middleware entre elastic y kibana, es decir, la parte de la recogida de muestras/eventos y la parte dónde se visualizan esas muestras. Logstash hace de filtro y de motor de normalización de datos entre los diferentes activos que tiene asignados para así poder tener todos los datos unificados según la especificación que queramos dar a cada uno de ellos.

Cada parte de la pila esta desarrollada en su propio lenguaje de desarrollo, siendo Java (o Groovy) el lenguaje de desarrollo de elastic, Ruby el de logstash y Javascript para Kibana. El único inconveniente que se ha podido observar es que la solución ELK está diseñada para trabajar más optimizada en entornos de cloud computing y no de manera local en un servidor. Dado que tendría que usar recursos propios de la máquina y conforme se vayan escalando nuevo recursos estos irán aumentando los de la máquina. Si hay limitación de hardware por los mismo puede llegarse a experimentar un cuello de botella entre elastic y kibana, siendo la carga de datos muy lenta y con tiempos de refresco bastante altos.

2.3. Sistemas de recolección y administración de eventos: SIEM

Un sistema de recolección y administración de eventos (SIEM), es una tecnología que se usa para la detección de amenazas y respuesta ante incidentes de seguridad a través de la obtención, en tiempo real o mediante un histórico, de eventos de seguridad a partir de una amplia variedad de fuentes o activos. Una de las principales características de un SIEM, es el gran alcance que tiene a la hora de recopilar una gran cantidad eventos para posteriormente correlar dichos eventos con alertas o incidencias de seguridad conocidas dentro del entorno corporativo.

2.3.1. Principales características

Estos son los puntos que principalmente gestiona un SIEM:

- Gestión de parches (actualizaciones) del kernel o software de terceros cómo adobe, java, etc.
- Antivirus en máquinas de usuarios o en servidor.
- Gestión de cortafuegos.

- Integración con Active Directory (LDAP).
- Sistema de prevención de intrusiones (en red: NIPS / basado en hosts: HIPS)
- Proxy / Filtro de contenidos
- Email: anti-spam / anti-phishing
- Análisis de vulnerabilidades
- Herramientas de seguridad opcionales:
  - ◇ IPS para redes Wifi.
  - ◇ Control de firewall web.
  - ◇ Aplicaciones que monitoricen bases de datos.
  - ◇ Prevención de pérdida de datos.
  - ◇ Gestión de riesgos y herramienta de cumplimiento de políticas

2.3.2. Análisis

Aunque esta herramienta sea cómo un conglomerado de aplicaciones o herramientas de detección y análisis, su puesta en marcha no es así tan fácil cómo cabría esperarse dado que cada despliegue requiere de unos activos o fuentes diferentes. Además, cada solución final requerirá de unas especificaciones distintas, con lo que su implantación depende en gran medida de la facilidad de adaptación al entorno y también de que los responsables de dicha herramienta tengan un profundo conocimiento sobre ella.

Una de las palabras más comunes que definen la implementación de un SIEM es: desalentador. A menudo termina costando más de lo previsto, requiere una experiencia sobre la herramienta que a menudo suele ser externalizada (sobre el propio fabricante) y puede llevar un tiempo considerable antes de obtener resultados tangibles.

Los motivos para los que generalmente se introducen un sistema cómo un SIEM en la red corporativa suelen ser varios, pero entre los más destacados suelen ser: el cumplimiento de una normativa industrial o de un gobierno, la gestión de incidentes recurrentes de seguridad o también que en la licitación de un contrato esté contemplado la puesta en marcha de este sistema para una mayor calidad del servicio prestado por un tercero o por la propia entidad. Y aún así, que la propia empresa ya gestione sus eventos internos o los monitorice, no implica que su migración al SIEM sea inmediata.

Además, la licitación o adquisición de un producto de estas características supone que el entorno sobre el que se va a aplicar contiene dispositivos de seguridad que monitorizar (sino se está realizando dicho control), que se dispongan de herramienta de centralización de datos (físicamente o en cloud) y que se quiera un sistema de gestión 24/7, incluido la monitorización fuera de horario laboral.

Son estas razones por las que a veces un sistema tan complejo y gigantesco puede que genere un sobrecoste o cubra pocas áreas de la red en las que no se tiene necesidad de vigilar. Siendo esto un incoveniente finalmente, dado que hay otras herramientas (de las que se nutre el SIEM) que ya cumplen con dicha funcionalidad a coste de tener un sistema muy pesado que gestionar.

2.4. Solución tecnológica de mi proyecto

La idea principal de mi proyecto surge fruto de la necesidad de monitorizar un red corporativa a través de un mecanismo de gestión automatizada de eventos, o lo que viene siendo un SIEM. Los pasos para la realización de este sistema se han modularizado y dividido en diferentes etapas que se acometarán cómo un todo dentro de un proyecto de investigación del grupo de ciberseguridad de la universidad de granada (UCyS - http://ucys.ugr.es/).

Dicho proyecto consistirá en la monitorización de la red corporativa de Mercagranada y todo lo que ello conlleva: sistemas informáticos, gestión de bases de datos, sistemas pci, sistemas perimetrales, etc.

[Explicar que es lo que diferencia mi pfc de las anteriores herramientas analizadas: lookwise, ELK y un SIEM. Hay otras herramientas cómo splunk o sumologic - http://blog.takipi.com/log-management-tools-face-off-splunk-vs-logstash-vs-sumo-logic/]

2.5. Tecnologías implementadas

A continuación explicaré las diferentes tecnologías/bibliotecas/lenguajes que se han empleado para la elaboración del proyecto y porque se han escogido por encima de otras posibles soluciones.

2.5.1. Python

Python es un lenguaje multiparadigma cuya estructura principal está integramente basada en objetos con sus diferentes métodos y atributos internos. Además también posee tipado dinámico, recolector/administrador de la memoria interna y un sistema de referenciado interno de atributos.

Las principales características que hicieron de su elección fueron:

- Es un lenguaje que facilita la implementación de una forma muy ágil y dinámica.
- Su sistema de paquetes es muy intuitivo y ligero. Puedes instalar cualquier dependencia que necesites descargandola del repositorio de paquetes PyPi en el que

se pueden encontrar infinidad de soluciones software dependiendo de lo que necesites. Sino, siempre se puede definir tú propio paquete software y subirlo al repositorio.

- Es de licencia similar a la BSD (Python Software Foundation License) compatible con la GNU GPL. Así que puedes modificar cualquier aspecto del kernel del lenguaje o mejorar cualquier funcionalidad que creas oportuna.

- Permite la fácil portabilidad entre plataformas, ya que sólo requiere de un intérprete que traduzca dicho código fuente a lenguaje máquina por lo que no existe una pesada fase de compilación por parte del sistema.

- Tiene una amplia comunidad de desarrolladores y es muy fácil de aprender en un corto periodo de tiempo para alguien iniciado.

- Es ampliamente usado en el mundo de la seguridad informática para ser usado en parsers, procesadores de eventos y usos con la web cómo principal reclamo (Protocolos, paquetes, tráfico, etc).

Python frente a otras soluciones

En la fase de estudio de tecnologías, se planteo la posibilidad de desarrollar el proyecto con el lenguaje de programación Java pero se descarto por los siguientes aspectos:

- Exige de unos conocimientos más avanzados sobre el control de procesos e hilos aunque sea más eficiente, también es más pesada la ejecución de los mismos en un sistema concurrente que puede que tenga muchos activos o fuentes que usar.

- Lenguaje fuertemente tipado y muy estructurado.

- Precisa de una compilación previa que pudiera incrementar los costes de empaquetar dicha solución y además solo se puede ejecutar en un entorno dónde exista una JVM o java virtual machine.

- Para adaptar la solución obtenida a un resultado web tendría que hacer uso de un framework o IDE que ayudase a la hora de la gestión y diseño de la interfaz web así cómo la comunicación. En Python frameworks cómo Django que facilitan la tarea del despliegue en formato web y lo más cercano que se le parece es el lenguaje Groovy sobre el que se basa el software Elasticsearch.

2.5.2. Procesos vs hilos

En el proceso de desarrollo del proyecto hubo varios momentos en los que se valoró y estudio las difererencias entre un desarrollo software basado en procesos, a la hora de cada nuevo input que llegase a la monitorización, frente hilos o threads. A continuación, se hará un breve resumen sobre las diferencias entre ambos:

Procesos

Un proceso, se basa en la parte de un programa que se ejecuta en nuestro sistema, es decir, un conjunto de recursos reservados del sistema.

Hilos

Un hilo o thread, es similar a un proceso dado que hace uso de unos recursos reservados del sistema. Pero con una gran diferencia, porque si los procesos ocupan diferentes espacios de memoria, los hilos comparten ese espacio entre ellos.

Problemas de los hilos

La normal general es que un conjunto de hilos o procesos tiendan a compartir información entre ellos. Por lo que la solución de los hilos a priori parece ser la más adecuada dado que compartir información será mucho más fácil. Sin embargo, la compartición de grandes cantidades de información y haciendo un uso amplio de tareas concurrentes, pueden producir dos situaciones delicadas: el bloqueo mutuo (deadblock) y la condición de carrera (race condition).

Hilos del kernel vs hilos del usuario

Diferentes hilos comparten un mismo PID mientras que diferentes procesos, poseen sus propios PID. Sin embargo, esto no sucede a nivel de kernel. Los hilos del kernel tienen sus propios PID, debido a la forma en la que el kernel es ejecutado.

El kernel (para sistemas GNU) en sí mismo no se ejecuta cómo un proceso sino que sus tareas se ejecutan cómo parte de otros procesos. Debido a la gran cantidad de tareas que se ejecutan, en el kernel se realiza una implementación o acción alternativa para operar de forma similar a los procesos (esto es a lo que se le conoce cómo demonios).

Solución que aplica Python

A la hora de la implementación para comprobar que hilos se están ejecutando a través el hilo padre, se utiliza el método de la clase Thread enumerate.

Lo que se realiza en esa comprobación es que dentro de la pila o lista de thread que se han lanzado haya alguna coincidencia de objetos de la clase Iptables y si la hay ya existe un hilo previo en ejecución que hace las comprobaciones pertinentes.

```
for threads in threading.enumerate():

    test = Iptables(
        args=(1,),
        source={'T': 'Firewall', 'M': 'iptables', 'P':
            '/var/log/iptables.log',
                'C': './secapp/kernel/conf/iptables-conf.conf'}
    )
    if type(threads) == type(test):
        exist_thread = True

if not exist_thread:
    thread_iptables = Iptables(
        args=(1,),
        source={'T': 'Firewall', 'M': 'iptables', 'P':
            '/var/log/iptables.log',
                'C': './secapp/kernel/conf/iptables-conf.conf'}
    )
    thread_iptables.start()
```

2.5.3. Django

Django es un framework web de alto nivel para el lenguaje de programación Python. Éste framework permite un despliegue de una aplicación de escritorio al entorno web de una forma sencilla, segura y fácilmente escalable.

Su metodología es MVT - Model View Template en la que se explicarán en que consisten cada una de ellas:

Model

Model o Modelo es la parte encargada de la gestión con la base de datos y de abstraer su uso encapsulandola mediante clases que representan a cada una de las instancias de la BD (o tablas).

View

View o Vista es la parte encargada de la gestión entre la base de datos y el template. Puede llevar a confusión esta metodología de desarrollo web, ya que existe una similar: MVC. Pero en MVC (Modelo-View-Controller) el modelo se encarga de la gestión de la BD, la vista de representar los datos y el controlador de administrador o motor de contenidos entre la BD y la vista final de la aplicación web.

Así pues la Vista en el framework Django se representa al modelo MVC cómo el controlador.

Template

Template o Plantilla se refiere a la parte encargad de reprensentar la información almacenada en BD y procesada y generada mediante la Vista (View). Al contrario de un modelo MVC, en dónde el controlador se encarga de generar las vistas, en este modelo se representan las mismas cómo una plantilla de muchas otras que se van enrutando a diferentes funciones generadas por la vista.

Dicho esto siempre habra una plantilla Master o padre de la que heredaran todas las hijas que vayamos asociando a nuestra web. Estas heredan una estructura o skeleton en formato html enriquecido con lenguaje propio de Django en formato python. Veamos un ejemplo de plantilla Master y una plantilla que se nutre de éste skeleton:

```
<!DOCTYPE html>
{% load staticfiles %}
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
        initial-scale=1, maximum-scale=1"/>
    <title>Security Sensor | {% block title %}{% endblock %}</title>

    <link rel="stylesheet" href="{% static 'css/main.css' %}"
        type="text/css"/>
    <link rel="stylesheet" href="{% static
        'bower_components/c3/c3.css' %}" type="text/css"/>

</head>
<body>

{% block content %}

{% endblock %}

    <script src="{% static 'bower_components/react/react.js'
        %}"></script>
    <script src="{% static 'bower_components/react/react-dom.js'
        %}"></script>
    <script src="{% static 'bower_components/babel-browser/browser.js'
        %}"></script>
    <script src="{% static
        'bower_components/jquery/dist/jquery.min.js' %}"></script>
    <script src="{% static 'bower_components/marked/marked.min.js'
        %}"></script>
    <script src="{% static 'bower_components/d3/d3.min.js' %}"
        charset="utf-8"></script>
    <script src="{% static 'bower_components/c3/c3.min.js'
        %}"></script>
    <script src="{% static
        'bower_components/react-d3/dist/public/js/react-d3.min.js'
        %}"></script>
```

En el fragmento de la plantilla Master incluimos los archivos estáticos del sistema (**load staticfiles**) y además especificamos dónde iría el cuerpo de las vistas o plantillas hijas que heredarán este skeleton mediante la especificación de un bloque (**block content** y **endblock**). Y ahora veremos un ejemplo de lo que podría ir dentro de ése bloque.

```
{% extends 'MasterPages/MasterPage.html' %}
{% block title %}Home{% endblock %}
{% block content %}

<div id="content">

</div>
<div id="chart">
<svg></svg>
</div>
<div id="sub-content">

</div>
<div id="events-info">

</div>
{% if latest_source_list %}
<ul>
    {% for logsource in latest_source_list %}
    <li><a href="{% url 'secapp:events' logsource.id %}">{{
        logsource.Type }}</a></li>
    {% endfor %}
</ul>
{% else %}
<p> No sources are available.</p>
{% endif %}
{% endblock %}
```

Aquí se observa perfectamente cómo se heredan las características de la plantilla Master y se define el bloque de contenido que en la plantilla Master ya se especifico.

2.5.4. Dependencias Python

2.5.5. C3js

2.5.6. D3js

2.5.7. LaTeX

2.5.8. Reactjs

2.5.9. Sqlite

2.5.10. Rsyslog

2.5.11. Logrotate

2.5.12. Syslog

2.5.13. Iptables

2.5.14. Django-rest

2.5.15. Json

2.5.16. Pycharm

2.5.17. Taiga

2.5.18. Bitbucket

2.5.19. Git

2.5.20. Digital Ocean

2.5.21. Nginx

Capítulo 3

Especificación y análisis de requisitos

Análisis de requisitos en el plan se necesita desarrollar una aplicación para solventar éste problema y se ha planteado con estas premisas (source, controller, manager, etc) que a su vez han necesitado de un aprendizaje, explicar porque se ha hecho una cosa de una manera (sqlite3 vs mysql), etc.

Especificacion de los requisitos
Análisis (Justificacion de la especificacion)

Capítulo 4

Diseño

<diseño>——</diseño>

- estructura de clases (diagrama)
- diseño de la vista (patrón)
- arquitectura del sistema

Capítulo 5

Implementación

<implementacion>——</implementacion>

- git
- archivos de instalación
- archivos de configuración (estructura)
- fuentes del código
- jerarquía de clases (implementación)

Capítulo 6

Evaluación

- pruebas funcionales y no funcionales
- caja blanca
- caja negra

Capítulo 7

Planificación y estimación de costes

Ya tengo hecho así por encima el diagrama de gantt de todo el proyecto. Tendría que definir las tareas en el diagrama pero lo que haré será meter aquí los apuntes que he ido metiendo en la herramienta Taiga para que se vea alguna descripción o apuntes que he ido tomando por cada tarea del diagrama.

7.1. Software utilizado

El lenguaje usado es python 2.7, luego django con todo los paquetes asociados (poner algún enlace de referencia dónde se listen los paquetes más importantes que he tenido que instalar sin nombrar todas las dependencias de estas)
He usado cómo IDE Pycharm del projecto jetbrains
Para la memoria LaTeXy para las gráficas tikz supongo.

7.2. Licencia

La licencia del proyecto para su posterior uso. En el actual repositorio de bitbucket no se ha especificado la licencia tal cuál, pero cuando ya este más estable el asunto también lo pondre en github para subirlo y demás. Ahí ya si que tendré que especificarla. En principio sino es para ningún propositio comercial, con la MIT sería suficiente. Sino alguna variante de la GPL o la Mozilla o similares.