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, 3 de julio 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).

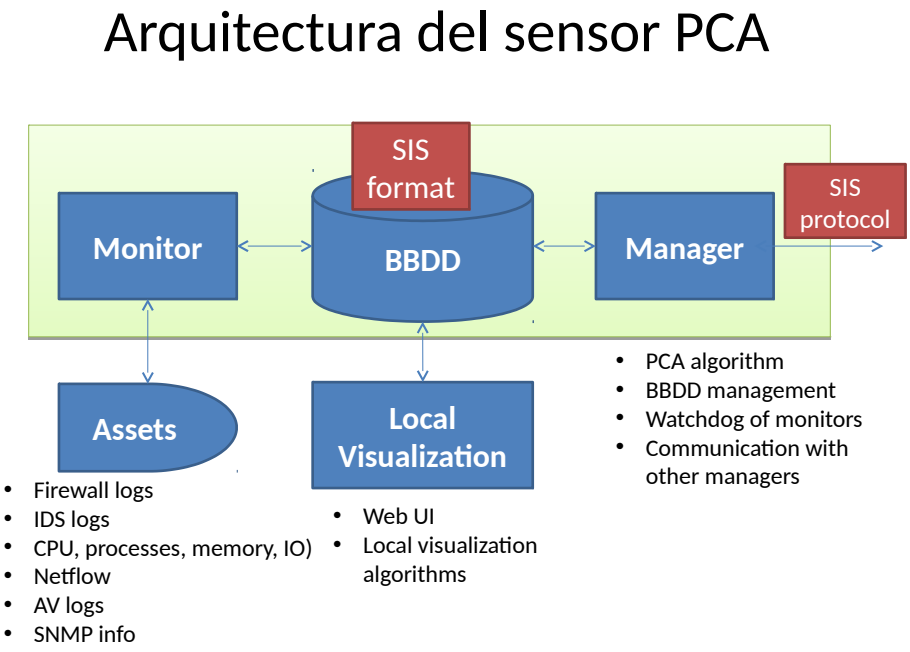


Figura 1.1: Arquitectura interna del software

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.

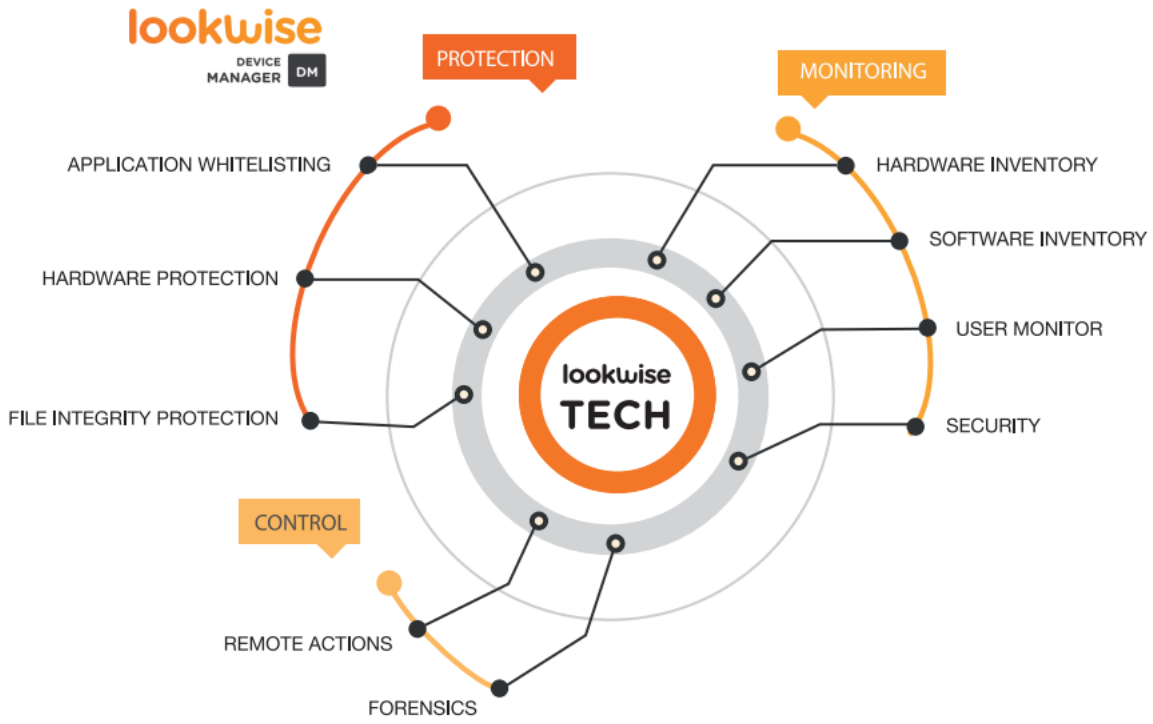


Figura 2.1: Funcionalidades Lookwise

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:

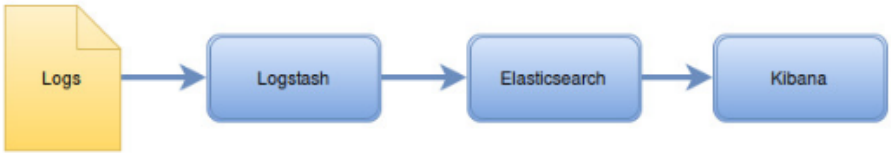


Figura 2.2: ELK Stack

- 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

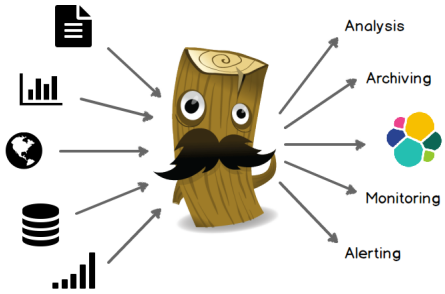


Figura 2.3: Logstash description

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.

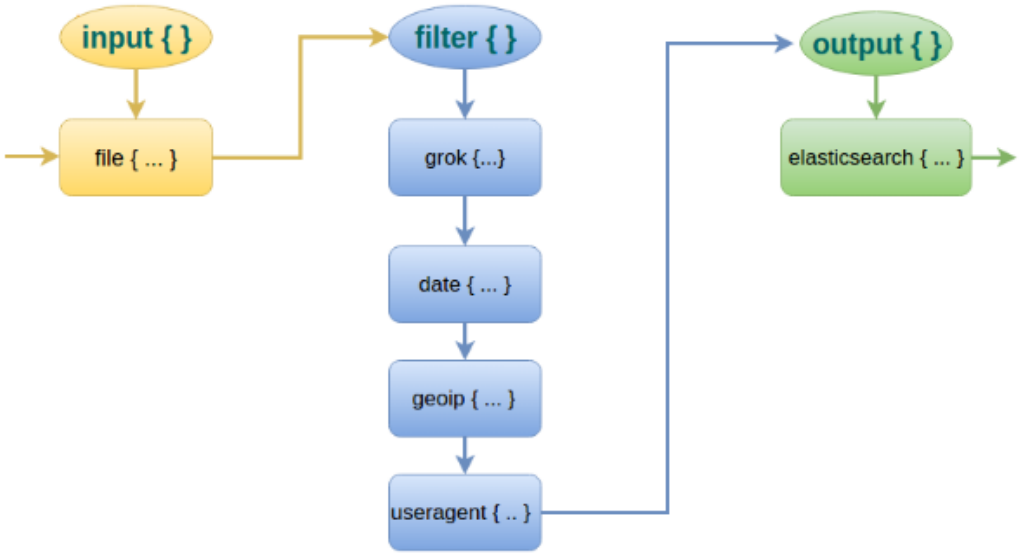


Figura 2.4: Diagrama de flujo interno de logstash

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 el diseño, implementación y evaluación de una metodología basada en el análisis multivariante para la detección y diagnóstico distribuido de anomalías en red, que permita:

- Detección temprana de vulnerabilidades o incidencias en dispositivos de monitorización.
- Gestión a nivel de capas, para el procesamiento y tratamiento entre los distintos niveles y su posterior comunicación con un gestor de amenazas cómo podría ser un SIEM.
- Empleo de algoritmos de análisis multivariante y su posterior compresión para envio entre las redes internas al SIEM.
- Visualización de las incidencias detectadas para su análisis por un ente humano.

La principal diferencia que existe entre mi proyecto final de carrera y las herramientas anteriormente analizadas es que el desarrollo y gestión de la información no se hace de forma analítica sino que se gestionan cantidades de datos (en la fase de recolección de logs de dispositivos) y estos no reciben ningún tipo de filtro o procesamiento previo. Se transportan desde el dispositivo monitorizado a otro en dónde se correlan los distintos logs o recolecciones según patrones definidos por los ingenieros de seguridad de la compañía encargada de detectar anomalías en los mismos.

No existe ningún tipo de mecanismo analítico de esta información dado que el punto de inflexión o la inteligencia esta en las etapas finales y no en las previas del tratamiento de la información. Para dotar de una mayor funcionalidad a estos sistemas hay extensiones o módulos que desarrollan las compañías que amplian la capacidad analítica de la misma por una cantidad económica bastante alta y su uso suele estar supeditado a licencia.

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

Web: https://www.python.org/

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

Web: https://www.djangoproject.com/

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:

```
1  <!DOCTYPE html>
2  {% load staticfiles %}
3  <html lang="en">
4  <head>
5      <meta charset="UTF-8">
6      <meta name="viewport" content="width=device-width,
           initial-scale=1, maximum-scale=1"/>
7      <title>Security Sensor | {% block title %}{% endblock %}</title>
8
9      <link rel="stylesheet" href="{% static 'css/main.css' %}"
           type="text/css"/>
10     <link rel="stylesheet" href="{% static
           'bower_components/c3/c3.css' %}" type="text/css"/>
11
12 </head>
13 <body>
14
15 {% block content %}
16
17 {% endblock %}
18
19     <script src="{% static 'bower_components/react/react.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 que ya se especifico (va dentro de las etiquetas **block** y **endblock**).

2.5.4. Dependencias Python

Estas son las dependencias que se necesitan para la fase de desarrollo del proyecto. Para poder uso de ellas, previamente tendremos que tener instalado un entorno virtual o virtualenv desde el cual hacer la instalación de estos paquetes.

Virtualenv

Es una herramienta que nos permite crear entornos aislados de Python. De esta forma podemos realizar pruebas de dependencias o paquetes para un entorno de desarrollo, en dónde las instalaciones o pruebas no afecten a las dependencias internas del sistema.

- Django 1.9.2: https://pypi.python.org/pypi/Django/1.9.2

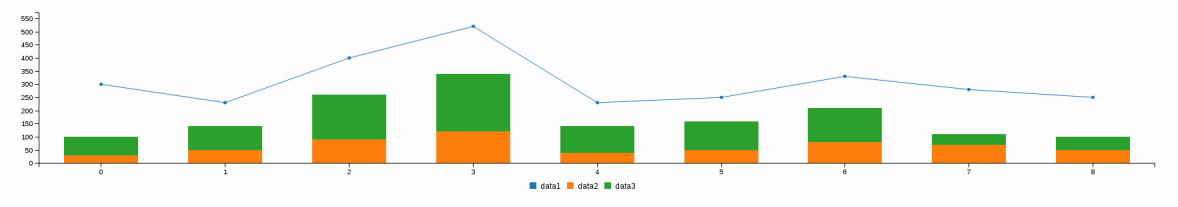
- argparse 1.4.0: https://docs.python.org/2.7/library/argparse.html
- dnspython 1.12.0: https://pypi.python.org/pypi/dnspython/1.12.0 - Se usa para hacer resolución directa e inversa de DNS.
- gunicorn 19.4.5: https://pypi.python.org/pypi/gunicorn - Sirve para desplegar un servidor WSGI (Web Server Gateway Interface) HTTP en entornos Unix.
- optional-django 0.3.0: https://pypi.python.org/pypi/optional-django/ - Otras utilidades del framework Django
- pygtail 0.6.1: https://pypi.python.org/pypi/pygtail - Sirve para leer archivos de log que no se han leido aún. Similar al comando tail -f en sistemas Unix.
- python-dateutil 2.4.2: https://pypi.python.org/pypi/python-dateutil/2.4.2 - Extensión al módulo principal de Python datetime.
- react 2.0.2: https://pypi.python.org/pypi/react/2.0.2 - Módulo que ejecuta un servidor de datos para los componentes react de la aplicación en Python.
- requests 2.9.1: https://pypi.python.org/pypi/requests/ - Bibloteca Python para poder consumir recursos HTTP de forma segura y controlada.
- setuptools 20.1.1: https://pypi.python.org/pypi/setuptools - Herramienta para la descarga, compilación, instalación, desinstalación y actualización de todos los paquetes Python. El resultado de la misma se puede usar mediante pip que se nutre el repositorio de paquetes PyPi.
- six 1.10.0: https://pypi.python.org/pypi/six - Es una bibloteca de compatibilidad entre Python 2 y 3.
- wheel 0.29.0: https://pypi.python.org/pypi/wheel - Es un gestor o generador de paquetes en Python.
- wsgiref 0.1.2: https://pypi.python.org/pypi/wsgiref - Es una biblioteca que sirve cómo soporte de validación para WSGI 1.0.1 para versiones de Python inferiores a la 3.2.
- configparser 3.5.0: https://docs.python.org/2/library/configparser.html - Es módulo o class que implementa un lenguaje muy sencillo para el parseo de archivos de configuración, siguiendo una estructura similar a cómo se describen y procesan los archivos INI en sistemas Microsoft Windows.

2.5.5. C3js

Web: http://c3js.org/

Es una biblioteca gráfica basada D3js, que es otra biblioteca visual en Javascript. En este caso, la biblioteca hace uso de la funcionalidad para gráficas de D3js adaptando su

motor a otras posibles soluciones posibles de implementación.



2.5.6. D3js

Web: https://d3js.org/

D3.js es una biblioteca javascript para la manipulación de documentos basado en datos. D3 nos ayuda a la hora de dar vida a los datos usando HTML, SVG y CSS de una forma sencilla y visualmente muy espectacular. Tiene su propio lenguaje que hace uso de las funcionalidades de Javascript por debajo y cuya comunidad es muy amplia en dónde podemos obtener multitud de ejemplos visuales que demuestran la capacidad de generación de gráficas y eventos de la biblioteca.

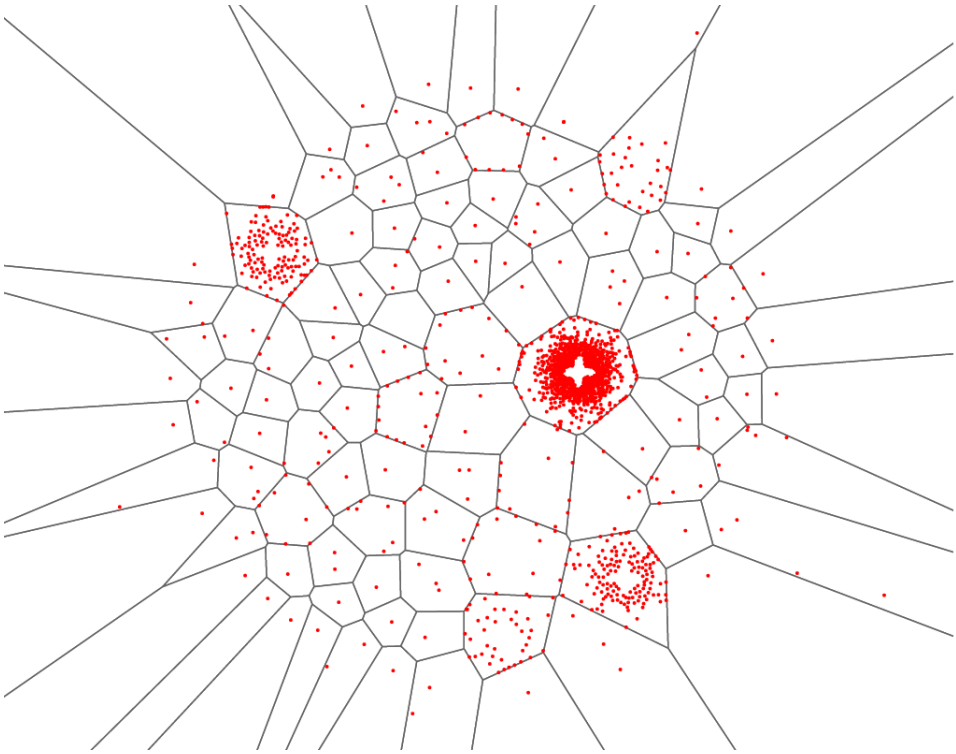


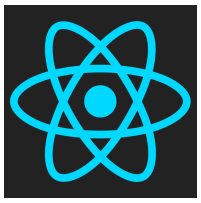
Figura 2.5: Ejemplo de D3js

2.5.7. LaTeX

Web: https://www.latex-project.org/

LaTeXes un lenguaje de marcado que sirve para la redacción de documentos científicos o técnicos. Con esta herramienta o lenguaje se ha desarrollado la memoria actual del proyecto de final de carrera.

2.5.8. Reactjs



Web: https://facebook.github.io/react/

React es una biblioteca Javascript que permite construir interfaces de usuario en nuestra aplicación web. Existen soluciones similares a React cómo podría ser JQuery, pero en este caso el proceso de creación de componentes visuales se hace muy intuitiva mediante clases internas que encapsulan el contenido que posteriormente se traducirá a código Javascript nativo.

Para la traducción React utiliza un componente llamada JSX, pero en nuestra solución se ha usado el paquete Babel (https://babeljs.io/) que es similar pero en formato Javascript y no NodeJS.

2.5.9. SQLite

Web: https://www.sqlite.org/

SQLite es un biblioteca software que implementa un motor para bases de datos SQL. Sus principales características son las siguientes:

- Las transacciones de datos son atómicas, consistentes, aisladas y durarderas (ACID) incluso después de que el sistema tengo un fallo o se apague inesperadamente.
- No necesita de una administración o configuración previa para su normal funcionamiento.
- Implementación de un sistema SQL integro con características más avanzadas cómo indexación parcial.

- La base de datos, en su totalidad, se almacena cómo un archivo normal en disco. Esto es útil para ser cargado directamente cómo un archivo en una aplicación.
- Soporta bases de datos de Terabytes de tamaño y Gigabytes de string o archivos.
- Facilidad de uso mediante su API interna.
- No tiene ninguna dependencia externa.
- Multiplataforma: Android, BSD, iOS, Linux, Mac, Solaris, VxWorks y Windows soportan este tipo de formato de base de datos.
- El código fuente está bajo dominio público para cualquier tipo de uso.

2.5.10. Rsyslog

Web: http://www.rsyslog.com/

Rsyslog (Rocket-Fast System for Log Processing), es un sistema de recogida de logs de sistemas UNIX (servidos mediante syslogd) que nos permite manipularlos y exportarlos a un formato más adecuado para su procesado.

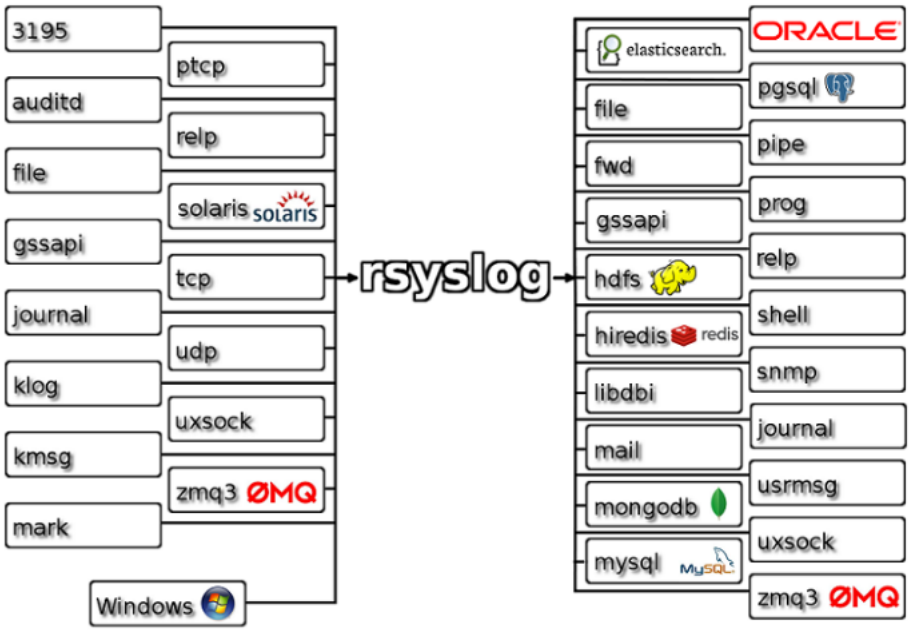


Figura 2.6: Esquema de Rsyslog

- Multihilo
- TCP, SSL, TLS RELP
- Se puede filtrar cualquier parte de los mensajes de syslog

- Se puede configurar el formato de salida de la recolección

2.5.11. Logrotate

Web: https://github.com/logrotate/logrotate
man: http://linux.die.net/man/8/logrotate

Es una utilidad de los sistemas UNIX que nos permite simplificar la administración de archivos de logs en un sistema en dónde haya logs de muchos tipos de fuentes. Nos permite automatizar la rotación, compresión, eliminación y envío por email de los archivos de logs del sistema. Logrotate se encuentra normalmente corriendo cómo un proceso cron diario.

```
/var/log/iptables.log
        {
                rotate 7
                daily
                missingok
                notifempty
                delaycompress
                compress
                postrotate
                        invoke-rc.d rsyslog restart > /dev/null
                endscript
        }
```

Figura 2.7: Configuración de iptables para Logrotate

Ahora se realizará una breve explicación de cada configuración que se ha establecido sobre el archivo iptables.log:

- **rotate <count>**: Los archivos de log son rotados una cantidad de <count>veces antes de ser eliminados o enviados por correo. Si <count>es 0, las versiones anteriores son eliminados antes de efectuarse la rotación.
- **daily**: Los archivos de log son rotados diariamente.
- **missingok**: Si el archivo de log no existe, ir al siguiente sin mostrar un mensaje de error.
- **notifempty**: No rotar el log si éste está vacío.
- **delaycompress**: Pospone la compresión de los logs previos para el siguiente ciclo de rotaciones. Esto sólo sucede cuando se combina con la opción **compress**.
- **compress**: Las versiones antiguas de logs son comprimidas con gzip por defecto.

- **postrotate-endscript**: Las líneas que se encuentran entre estas dos palabras en el configuración, son ejecutadas (usando /bin/sh) antes de que el archivo de log sea rotado y sólo si el log actual va a ser rotado.

  En este caso, el comando que se ejecuta fuerza a rsyslog a reabrir el archivo de log para escribir en él. Se suele usar después que logrotate mueva los archivos de logs antiguos, entonces rsyslog comienza a escribir en los nuevos.

2.5.12. Syslog

Web (man): http://linux.die.net/man/3/syslog

Sirve para el envío de mensajes al sistema de logs interno. syslog() genera un mensaje de log, el cuál se distribuye mediante el demonio syslogd.

2.5.13. Iptables

Web: http://www.netfilter.org/projects/iptables/index.html
Manual: http://www.netfilter.org/documentation/HOWTO/es/packet-filtering-HOWTO-7.html
man: http://linux.die.net/man/8/iptables

Herramienta de filtrado de paquetes IPv4/IPv6 y NAT. Iptables es usado para configurar, mantener e inspeccionar tablas de paquetes IP mediante filtros o reglas en el kernel de Linux. Se pueden definir multitud de tablas, en las que cada una pueda contener un número de reglas precargadas o que pueden ser redefinidas por el usuario.

2.5.14. Django-rest

Web: http://www.django-rest-framework.org/
Django REST es un framework potente y flexible que permite construir Web APIs. Se usa cómo un complemento al framework Django para el uso de API Restfull dentro de la propia aplicación.

2.5.15. JSON

Web: http://www.json.org/

JSON (Javascript Object Notation) es un formato de intercambio de datos ligero. Se usa para facilitar la legibilidad y escritura para los humanos y además es fácil de interpretar y parsear para una máquina. Está basado cómo un subconjunto de JavaScript y el Standard ECMA-262 3ª Edición.

JSON está construido sobre dos estructuras:

- Una colección de pares nombre/valores. En varios lenguajes, esto se realiza mediante objetos, registros, estructuras, diccionarios, tablas hash, listas de claves o arrays asociativos.
- Una lista de valores ordenados. En la mayoría de lenguajes, esto se realiza mediante un array, un vector, lista o secuencia.

2.5.16. PyCharm

Web: https://www.jetbrains.com/pycharm/

PyCharm es un IDE que permite el trabajo con aplicaciones Python en sus respectivos entornos virtuales (virtualenv) así cómo la integración con frameworks de desarrollo cómo es el caso de Django. La aplicación en sí fue desarrollada nativamente mediante un editor de texto (Emacs) pero a la hora de realizar un empaquetado e integración con otra herramientas se optó por éste IDE.

2.5.17. Taiga

Web: https://taiga.io/

Taiga es un gestor de proyectos que nos permite implementar una metodología SCRUM o Kanban sobre nuestro proyecto. Es una herramienta muy intuitiva y colaborativa, que permite definir hitos/tareas/wikis para solucionar cada punto de la fase de desarrollo de un proyecto. Además, permite la integración (WebHooks) con otras plataformas de repositorios de proyectos cómo GitHub, GitLab o BitBucket.

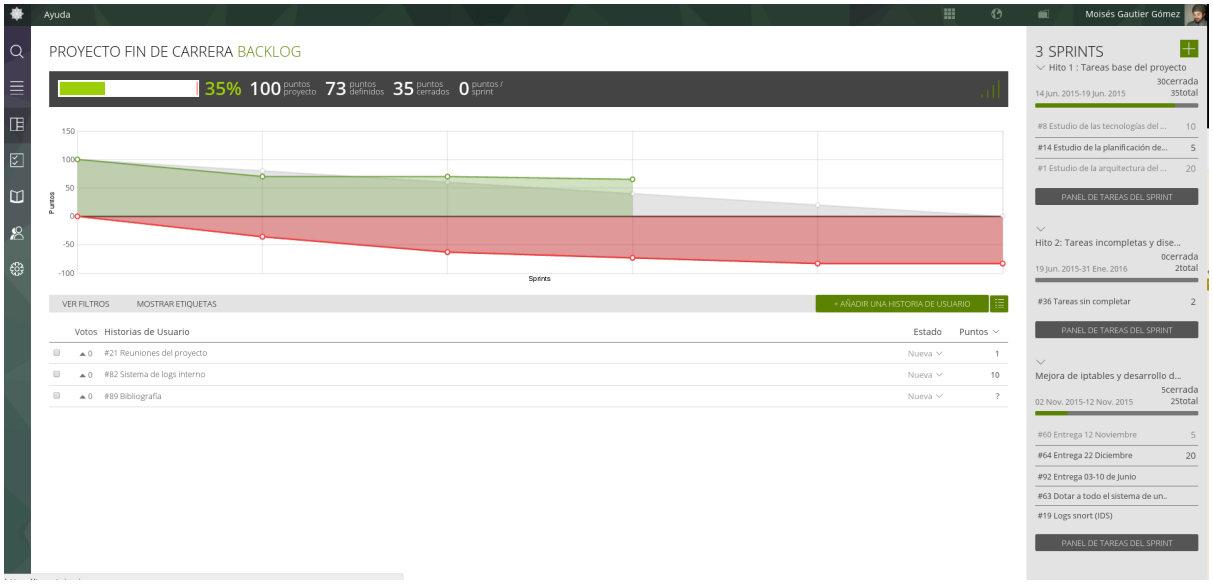


Figura 2.8: Backlog del proyecto siguiendo un SCRUM

2.5.18. Bitbucket

Web: https://bitbucket.org/
Repositorio: https://bitbucket.org/MGautierGomez/securityproject

Gestor de repositorios Git y Mercurial. Se optó por éste gestor para probar su funcionamiento, así cómo la posibilidad de tener repositorios privados para desarrollar partes del proyecto que necesitasen ser ocultadas.

2.5.19. Git

Web: https://git-scm.com/

Git es un sistema open-source de control de versiones diseñado manejar integramente las fases de desarrollo de proyectos, simples y complejos, con velocidad y eficiencia.

2.5.20. Digital Ocean

Web: https://www.digitalocean.com/

Servidor web para alojar proyectos en cloud. La ventaja de este servicio de VPS es que te permite desplegar máquinas de cualquier tipo (siempre que sean software lire) de una manera muy fácil y rápida. Además tiene un punto fuerte y es que la información se almacena en discos SSD, con lo que el procesamiento se ve muy mejorado a la hora de computar (en este caso eventos de Iptables).

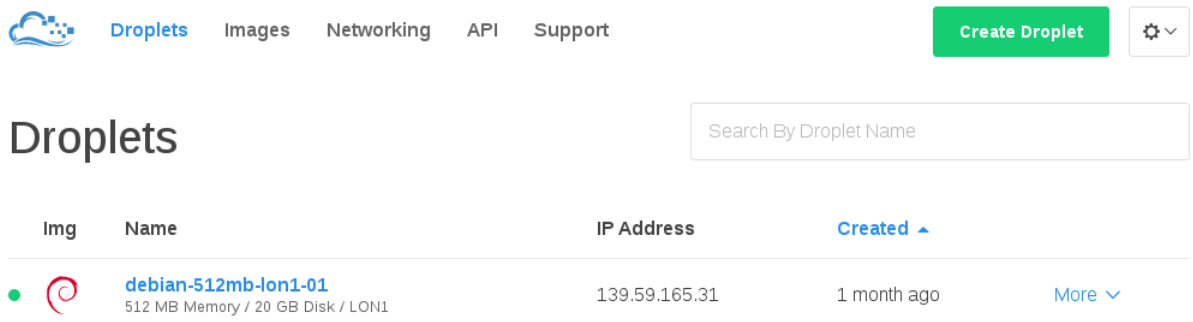


Figura 2.9: Droplet desplegado en digital ocean

2.5.21. Nginx

Web: http://nginx.org/
Características: http://nginx.org/en/

Nginx (engine x) es un servidor HTTP y proxy inverso, un servidor proxy de email y un servidor proxy genérico de TCP/UDP. Algunas de sus principales características son las siguientes:

- Sirve archivos estáticos, index y autoindexados.
- Acelera el proceso de proxy inverso con caché: Tolerancia a fallos y carga balanceada de datos.
- Soporta aceleración con cacheo de FastCGI, uwsgi, SCGI y servidores memcached: Tolerancia a fallos y carga balanceada de datos.
- Soporta SSL y TLS SNI.
- Soporta HTTP/2 con dependencia basada en prioridad y balanceo.

Configuración de nginx para el servidor en digital ocean:

```
server {

        root /var/www/html;

        # Tipos de archivos index de nuestro sistema

        index index.html index.htm index.nginx-debian.html;

        # Nombre del servidor en local

        server_name 139.59.165.31;

        location /static/ {
                alias ~/trunk/version-1-0/webapp/secproject/static/;
                expires 30d;
        }

        location / {
                proxy_set_header X-Forwarded-For
                   $proxy_add_x_forwarded_for;
                proxy_set_header Host $http_host;
                proxy_redirect off;
                proxy_pass http://127.0.0.1:8000;
                proxy_pass_header Server;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_connect_timeout 10;
                proxy_read_timeout 10;


        }
}
```

Figura 2.10: Configuración de nginx en digital ocean

Capítulo 3

Especificación y análisis de requisitos

Los requisitos funcionales y especificación de los mismos son los que a continuación se decriben.
Actualmente existen muchas tecnologías de monitorización de redes que a su vez tiene formato hardware, cómo controladores de sesiones de navegación de usuarios, firewall, proxys que enrutan el tráfico de una compañía hacia las redes externas, autenticación en sistemas VPN para acceso seguro sobre máquinas remotas, etc. El principal problema de estas tecnologías es que cada una pertenece a un ámbito distinto dentro del mismo departamento, es decir, pertenecen a las redes y a tráfico o flujo de comunicaciones pero se enmarcan en diferentes aspectos de cada uno de estos.

Para el caso que nos concierte en el proyecto, dentro del marco de investigación que define la totalidad de la infraestructura, la funcionalidad principal del mismo será:

- Definir los pasos para obtener recolección de logs de una fuente de seguridad. Configuración de rsyslog, logrotate y supervisord (este último en el caso de que sea necesario).

- Una vez configurada la máquina, configurar la instalación para cada fuente en particular. El ámbito del proyecto se enfoca sobre iptables.

- Realizar un sistema de parseo de logs para extraer la información necesaria para cada evento que se registre en el sistema.

- Extracción de características de dichos eventos y procesamiento mediante un algoritmo de frecuencias para detectar las anomalías más comunes.

- Sistema de gestión de base de datos en dónde se encuentren los datos en crudo recogidos, los procesados y los dispuestos para su visualización.

- Panel de control dónde visualizar la información de ese nodo con total detalle de la información.

- Compresión y envio de los datos a la capa superior dónde se alojará el SIEM.

3.1. Configuración local para procesamiento

Los primeros pasos la la obtención de logs o eventos generados por la fuente de seguridad, iptables, serán los de configurar al sistema interno de correlación de logs rsyslog junto con el sistema de rotacion de logs logrotate. Para el caso de rsyslog tenemos que definir un filtro para que cualquier evento que genere el sistema con un determinado mensaje definido en las reglas de iptables, sea capturado y almacenado en un determinado directorio. También hemos dotado a los logs del sistema de timestamp con mayor precisión para poder diferenciar eventos con mayor afinamiento y cambiado la tupla de permisos a la hora de crear un archivo con **FileCreateMode**.

```
1     #
2     # Use traditional timestamp format.
3     # To enable high precision timestamps, comment out the following
          line.
4     #
5     #$ActionFileDefaultTemplate RSYSLOG_TraditionalFileFormat
6
7     #
8     # Set the default permissions for all log files.
9     #
10    $FileOwner root
11    $FileGroup adm
12    $FileCreateMode 0644
13    $DirCreateMode 0755
14    $Umask 0022
15
16    # IPTABLES
17
18    :msg,contains,"IPTMSG= " -/var/log/iptables.log
19    :msg,regex,"^\[ *[0-9]*\.[0-9]*\] IPTMSG= " -/var/log/iptables.log
20    :msg,contains,"IPTMSG= " ~
```

Figura 3.1: Configuración de iptables para Rsyslog

También hemos de configurar Logrotate (más información sobre los campos visitar sección 2.5.11):

```
/var/log/iptables.log
        {
                rotate 7
                daily
                missingok
                notifempty
                delaycompress
                compress
                postrotate
                        invoke-rc.d rsyslog restart > /dev/null
                endscript
        }
```

Figura 3.2: Configuración de iptables para Logrotate

Además tenemos que definir una regla específica para el demonio de rsyslog en la cual se filtre también por los campos que queramos de iptables:

```
# into separate file and stop their further processing
if  ($syslogfacility-text == 'kern') and \\
($msg contains 'IPTMSG=' and $msg contains 'IN=') \\
then    -/var/log/iptables.log
    &   ~
```

Figura 3.3: Configuración de iptables.conf para Rsyslog.d

Estos tres paso o configuraciones nos permiten redireccionar un log de iptables al directorio **/var/log/** y en conreto para el archivo **iptables.log**. Esta configuración entrará en efecto una vez hayamos reiniciado el servico rsyslog o en el próximo inicio de sesión sobre la máquina.

3.2. Recogida y almacenamiento de logs

Para el caso que nos ocupa, iptables tiene una forma de definir reglas internas para el filtrado de paquetes según el tipo de comunicación que se establezca contra la máquina o desde la máquina hacia el exterior.

```
iptables -A INPUT -p tcp -m tcp --dport 22 -j LOG --log-prefix
   "IPTMSG=Connection SSH "
```

Figura 3.4: Ejemplo de regla iptables

En la siguiente sección explicaremos en profundidad cada una de las opciones de la regla, pero a simple vista podemos observar que el mensaje asociado a la regla coincide con la palabra clave del filtro empleado en rsyslog: **IPTMSG=**. Así pues, una vez dicho evento se genere el sistema y syslog lo procese como un mensaje de un servicio determinado, en este caso iptables, rsyslog filtrará dicho mensaje según su configuración para almacenarlo posteriormente en **/var/log/iptables.log**.

3.3. Iptables

El servicio de firewall del kernel de GNU Linux, iptables, nos proporciona una interfaz de reglas y tablas en dónde podemos definir patrones o reglas que actuen sobre el tráfico que llega o sale desde nuestra máquina. Las reglas que se han definido por defecto son las siguientes (si queremos más filtros hay que implicar al protocolo y su puerto asociado con un mensaje):

```
# Generated by iptables-save v1.4.21 on Mon Jan 25 20:37:18 2016
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -d 127.0.0.1/32 -p icmp -m icmp --icmp-type 8 -m state
   --state NEW,RELATED,ESTABLISHED -j LOG --log-prefix
   "IPTMSG=Connection ICMP "
-A INPUT -d 127.0.0.1/32 -p icmp -m icmp --icmp-type 8 -m state
   --state NEW,RELATED,ESTABLISHED -j DROP
-A INPUT -p tcp -m tcp --dport 22 -j LOG --log-prefix
   "IPTMSG=Connection SSH "
-A INPUT -p tcp -m tcp --dport 22 -j DROP
COMMIT
```

Figura 3.5: Configuracion reglas iptables

A continuación una breve explicación de cada opción o comando:

- -A: Añadir una nueva regla a una cadena de la tabla.

- -d: Especificación para la ip destino, en este caso localhost con una máscara de subred determinada.
- –dport: Especificación para el puerto destino al que se realizará la posible conexión o envío de paquetes.
- -p: Especificación del protocolo del paquete.
- -m: Especificación de matching, en este caso icmp o tcp dentro de la descripción del paquete.
- –icmp-type: Extensión del tipo de ping que se va a procesar desde la regla.
- –state: Tipo de paquete según conexión:
  - ◇ NEW: Paquete que crea una nueva conexión.
  - ◇ RELATED: Paquete que está relacionado a una conexión existente, pero que no es parte de ella, cómo un error ICMP o, un paquete que establece una conexión de datos FTP.
  - ◇ ESTABLISHED: Paquete que pertenece a una conexión existente (que tuvo paquetes de respuesta).
  - ◇ INVALID (no usado): Paquete que no pudo ser identificado por alguna razón: quedarse sin memoria o errores ICMP que no corresponden a ninguna conexión conocida. Normalmente estos paquetes deben ser descartados.
- -j: Acción de salto cuando encuentre dicha regla de paquetes. Se especifican dos acciones:
  - ◇ LOG –log-prefix “message” : Cuando se encuentre dicha regla se recolecta cómo log de la misma adjuntando un mensaje para diferenciarla del resto de reglas.
  - ◇ DROP : Cuando se encuentre dicha regla se descarta el paquete dentro del propio sistema. Al ir seguido LOG de un DROP el paquete se muestra en el registro de syslog para posteriormente ser eliminado del registro de almacenamiento de paquetes (Para esto usamos rsyslog que se encarga de almacenar dicho mensaje en un archivo de log).

Así pues una vez tengamos un evento o paquete o log de iptables en nuestro sistema generados mediante ssh 127.0.0.1 o ping 127.0.0.1 obtendremos lo siguiente:

```
1 [dom jul  3 17:04:03 2016] IPTMSG=Connection SSH IN=lo OUT=
    MAC=00:00:00:00:00:00:00:00:00:00:00:08:00 SRC=127.0.0.1
    DST=127.0.0.1 LEN=60 TOS=0x00 PREC=0x00 TTL=64 ID=39454 DF
    PROTO=TCP SPT=47706 DPT=22 WINDOW=43690 RES=0x00 SYN URGP=0
2 [dom jul  3 17:04:05 2016] IPTMSG=Connection SSH IN=lo OUT=
    MAC=00:00:00:00:00:00:00:00:00:00:00:08:00 SRC=127.0.0.1
    DST=127.0.0.1 LEN=60 TOS=0x00 PREC=0x00 TTL=64 ID=39455 DF
    PROTO=TCP SPT=47706 DPT=22 WINDOW=43690 RES=0x00 SYN URGP=0
```

Figura 3.6: Evento de ssh localhost en el sistema

Lo anterior correspondía a la salida del comando **$ dmesg -T** que muestra todos los mensajes del sistema que se han pasado al syslog. La opción -T se usa para especificar el timestamp de cada mensaje con una mayor precisión.

```
2016-07-03T17:03:35.664324+02:00 debian kernel: [23337.363387]
   IPTMSG=Connection SSH IN=lo OUT=
   MAC=00:00:00:00:00:00:00:00:00:00:00:00:08:00 SRC=127.0.0.1
   DST=127.0.0.1 LEN=60 TOS=0x00 PREC=0x00 TTL=64 ID=39454 DF
   PROTO=TCP SPT=47706 DPT=22 WINDOW=43690 RES=0x00 SYN URGP=0
2016-07-03T17:03:37.668326+02:00 debian kernel: [23339.369692]
   IPTMSG=Connection SSH IN=lo OUT=
   MAC=00:00:00:00:00:00:00:00:00:00:00:00:08:00 SRC=127.0.0.1
   DST=127.0.0.1 LEN=60 TOS=0x00 PREC=0x00 TTL=64 ID=39455 DF
   PROTO=TCP SPT=47706 DPT=22 WINDOW=43690 RES=0x00 SYN URGP=0
```

Figura 3.7: Log capturado y almacenado por rsyslog en /var/log/iptables.log

Lo anterior corresponde con la manipulación por parte de rsyslog del mensaje obtenido en syslog. Cómo podemos observar se añade un campo de timestamp de mayor precisión según la configuración que hemos establecido en rsyslog.conf para poder diferenciar con mayor exactitud eventos entre diferentes franjas de tiempo.

```
++++++++++++++++++++++++++++++++++++++++++++++++++
--------------------------------------------------

 Procesando linea --> 2016-07-03T17:12:53.632264+02:00 debian kernel:
    [23896.003739] IPTMSG=Connection ICMP IN=lo OUT=
    MAC=00:00:00:00:00:00:00:00:00:00:00:00:08:00 SRC=127.0.0.1
    DST=127.0.0.1 LEN=84 TOS=0x00 PREC=0x00 TTL=64 ID=64157 DF
    PROTO=ICMP TYPE=8 CODE=0 ID=14177 SEQ=7

--------------------------------------------------
++++++++++++++++++++++++++++++++++++++++++++++++++
---> Insertado registro: {'TAG': 'Connection ICMP', 'ID_Source_PORT':
   None, 'Protocol': u'ICMP', 'RAW_Info':
   '2016-07-03T17:12:53.632264+02:00 debian kernel 23896.003739
   IPTMSG=Connection ICMP IN=lo OUT
   MAC=00:00:00:00:00:00:00:00:00:00:00:00:08:00 SRC=127.0.0.1
   DST=127.0.0.1 LEN=84 TOS=0x00 PREC=0x00 TTL=64 ID=64157 DF
   PROTO=ICMP TYPE=8 CODE=0 ID=14177 SEQ=7 ', 'ID_Source_MAC': <Macs:
   00:00:00:00:00:00:00:00:00:00:00:00:08:00>, 'ID_Source_IP': <Ips:
   127.0.0.1>, 'ID_Dest_IP': <Ips: 127.0.0.1>, 'ID_Dest_PORT': None,
   'ID_Dest_MAC': <Macs: 00:00:00:00:00:00:00:00:00:00:00:00:08:00>}

++++++++++++++++++++++++++++++++++++++++++++++++++
---> Fin de procesado de linea
```

Figura 3.8: Procesamiento del log capturado y almacenado en la bd interna de la aplicación

3.4. Transformación de log en información útil: Parser

Una vez hemos descrito los pasos a seguir para obtener un log para un evento iptables, llega la hora de procesar y hacer útil todos esos datos que tenemos. Para esta finalidad tenemos que usar expresiones regulares para generar un parser con el que ser capaz de traducir todos esos datos en información útil para nuestra aplicación.

```
obj = Pygtail("/var/log/iptables.log")

while True:
    try:
        for line in obj:
            if len(line) > 1:
                self.process_line(line)
    except Exception as ex:
        print "Pygtail processing -> ", ex
```

Figura 3.9: Instancia de la clase Pygtail y lectura de las líneas del log

Hacemos uso del módulo o paquete Pygtail que nos permite leer archivos de log que aún no han sido leídos. Es una especie de **$ tail -f** a un archivo en concreto, pero usa un concepto de offset e inode para saber la última actualización y posición del archivo antes y después de ser abierto para así saber que parte de la última lectura se quedo en ejecución. Éste es otro punto importante dado que para hacer uso de esta funcionalidad debemos crear un archivo con el nombre del log, véase **/var/log/iptables.log** cuya extensión final sea offset (**/var/log/iptables.log.offset**) y sus permisos los siguientes:

```
    -rw-r--r-- 1 root adm 10391 jul  3 17:12 /var/log/iptables.log
    -rw-r--rw- 1 root root 14 jul  3 17:13 /var/log/iptables.log.offset
```

Figura 3.10: Permisos de los archivos iptables.log e iptables.log.offset

Una vez tengamos el archivo abierto y con sus líneas procesadas por Pygtail, es el momento de usar regex sobre los objetos string de cada línea de log. Para ello hacemos uso del método split para dividir por palabras, es decir, posiciones separadas en una lista a todas las coincidencias con una palabra que pudiera tener toda la cadena.

```
# Se trocea por palabras el log leido
line = re.split("\W? ", line)
```

Figura 3.11: Uso del método split sobre la entrada de lineas de log

Una vez separado por palabras toda la línea de log, pasamos a diferenciar entre las etiquetas que iptables pone a cada campo con su valor, es decir, una tupla key=>value.

```
tag_str = ((re.compile('^(.*)=')).search(str(line))).group(0)
tag_split = tag_str.split(',')
```

Figura 3.12: Obtenemos la tupla key=>value para cada etiqueta del log

Ya tenemos todas las etiquetas de los campos del log y ahora nos toca extraer su valor asociado para asignarlo al ORM de la base de datos, es decir, almacenar la información en la BD.

```
db_column = ['ID_Source_IP', 'ID_Dest_IP',
             'ID_Source_PORT', 'ID_Dest_PORT',
             'Protocol','ID_Source_MAC',
             'ID_Dest_MAC']

# El nombre de las tags, segun el orden de la
# columnas en db_column, las extraigo del fichero
# de configuracion a traves del registro info_config_file

labels = [self.info_config_file["Source_Ip"],
          self.info_config_file["Dest_Ip"],
          self.info_config_file["Source_Port"],
          self.info_config_file["Dest_Port"],
          self.info_config_file["Protocol"]]

# Almacenamos las etiquetas o campos del log de iptables
for it in tag_split:
    if len(it.split('=')) == 2:
        self.tag_log.append((it.split('='))[0].strip('\' '))

# Buscamos la correlacion entre los campos definidos en la
# configuracion con los extraidos del log de iptables

for label in labels:
    if (re.compile(label)).search(tag_str):
        if self.tag_log.index(label) > 0:
            db_column_name = db_column[0]
            register[db_column.pop(0)] = self.regexp(db_column_name,
               label, str(line))
            self.tag_log.remove(label)
    else:
        register[db_column.pop(0)] = None
```

Figura 3.13: Vamos asignando cada etiqueta y su valor a su asociado del ORM

Los siguientes pasos de comprobación de integridad de valores y demás se relegan a la visualización del interesado en el método process_line del archivo código fuente iptables.py

3.5. Workflow

3.6. Extracción de características

3.7. Visualización de eventos

3.8. Compresión

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.