Sensor para recopilación y visualización de información de seguridad en nodos de una red

Alumno: Moisés Gautier Gómez

moisesgautiergomez@correo.ugr.es

Director: Gabriel Maciá Fernández
Titulación: Ingeniería en Informática

Departamento de Teoría de la Señal, Telemática y Comunicaciones

Universidad de Granada
Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Motivación y objetivos

El objetivo principal del proyecto es desarrollar un software que permita recopilar y visualizar la información generada por las aplicaciones de monitorización y control de seguridad que se ejecutan en una máquina.

La motivación del mismo surge fruto de la necesidad de monitorizar una red corporativa a través de un mecanismo de gestión automatizada de eventos. Los pasos para la realización de este sistema se han modularizado y dividido en diferentes etapas que se acometarán como un todo dentro del proyecto de investigación VERITAS (http://nesg.ugr.es/veritas/) del Network Engineering & Security Group (NESG - http://nesg.ugr.es/) perteneciente al área de Ingeniería Telemática de la Universidad de Granada.

VERITAS

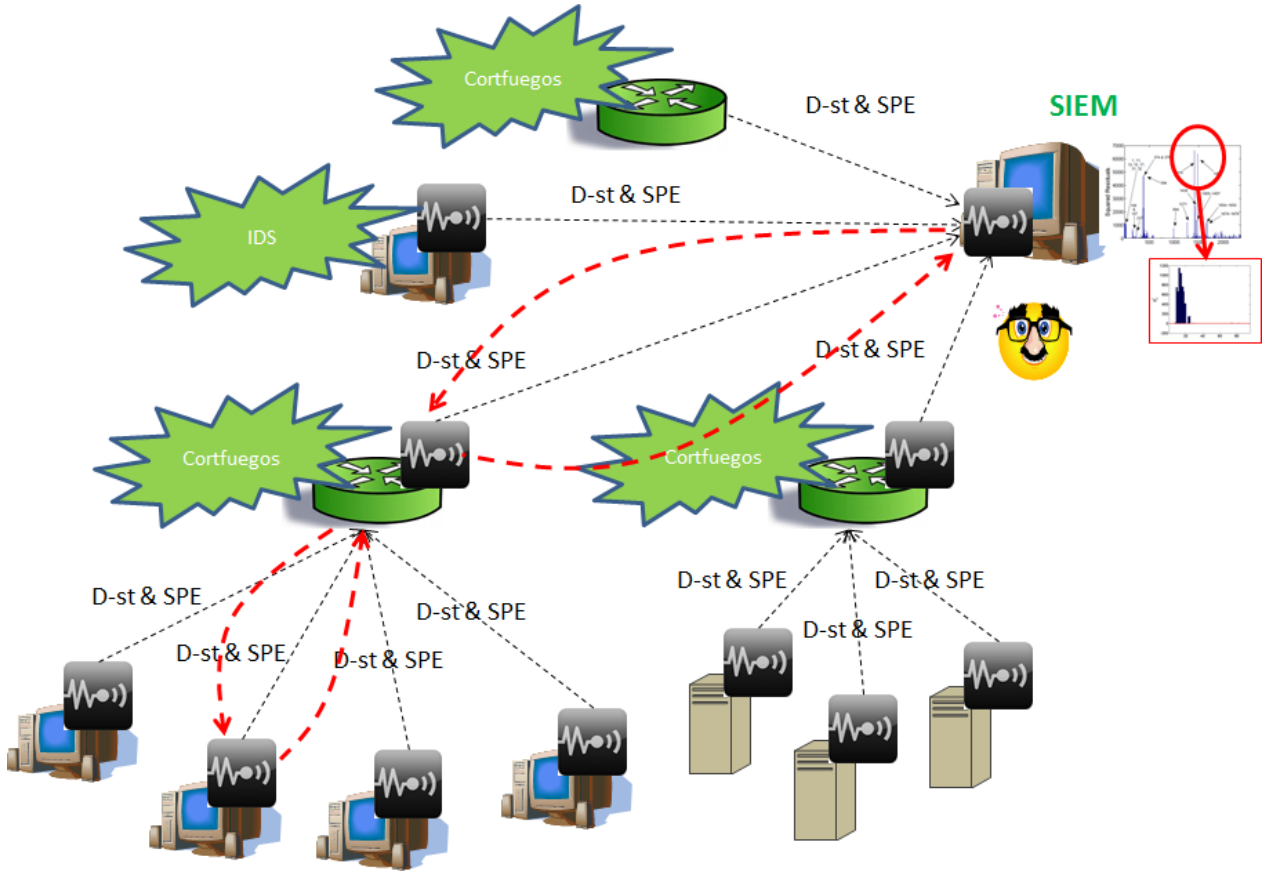
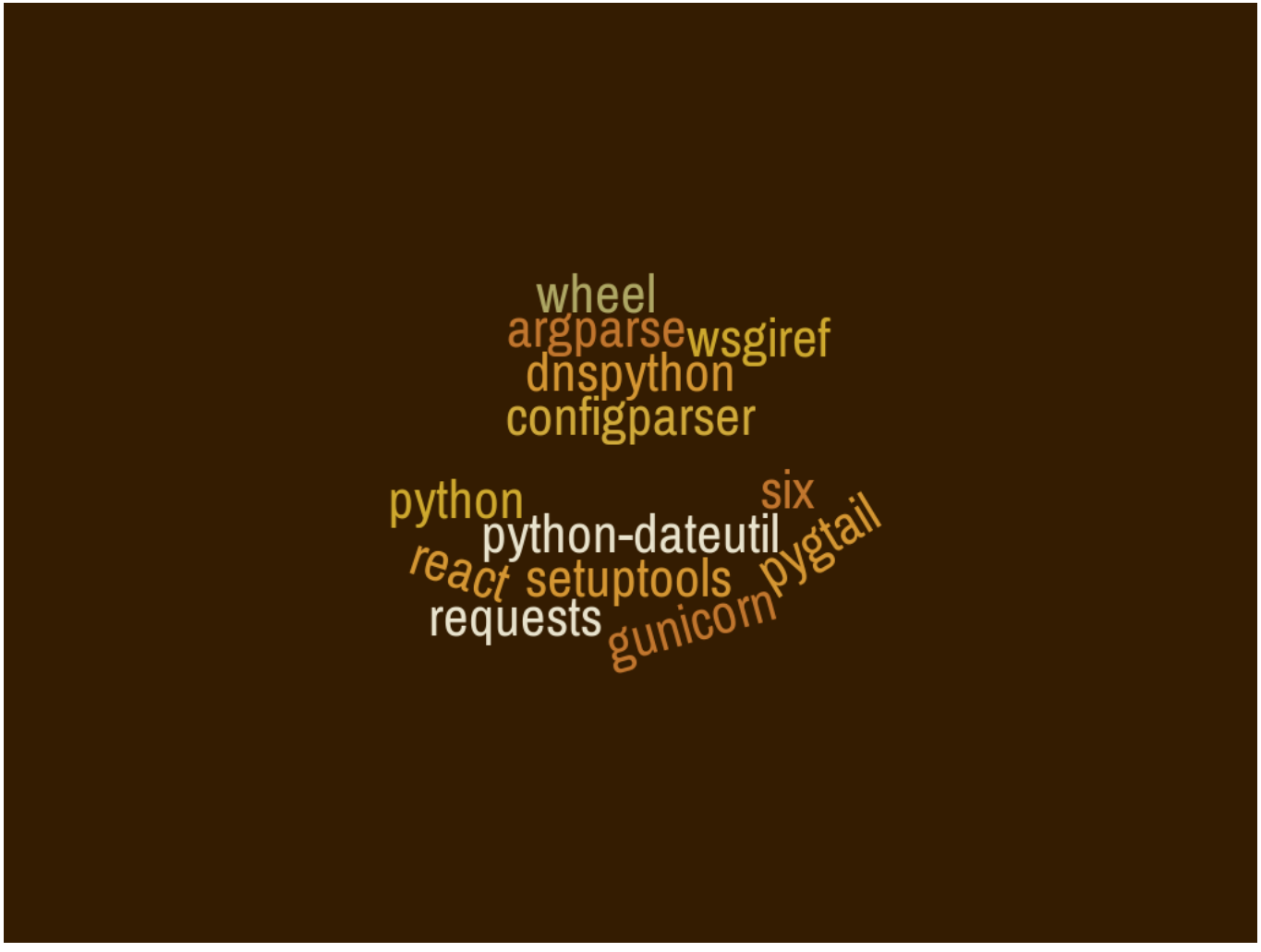


Figura - Nodos de una red - VERITAS

Desarrollo Backend



Desarrollo interfaz web

Gestión de proyecto

Servicios web

Eventos y recolección

Bibliotecas gráficas / javascript

Clases del sistema

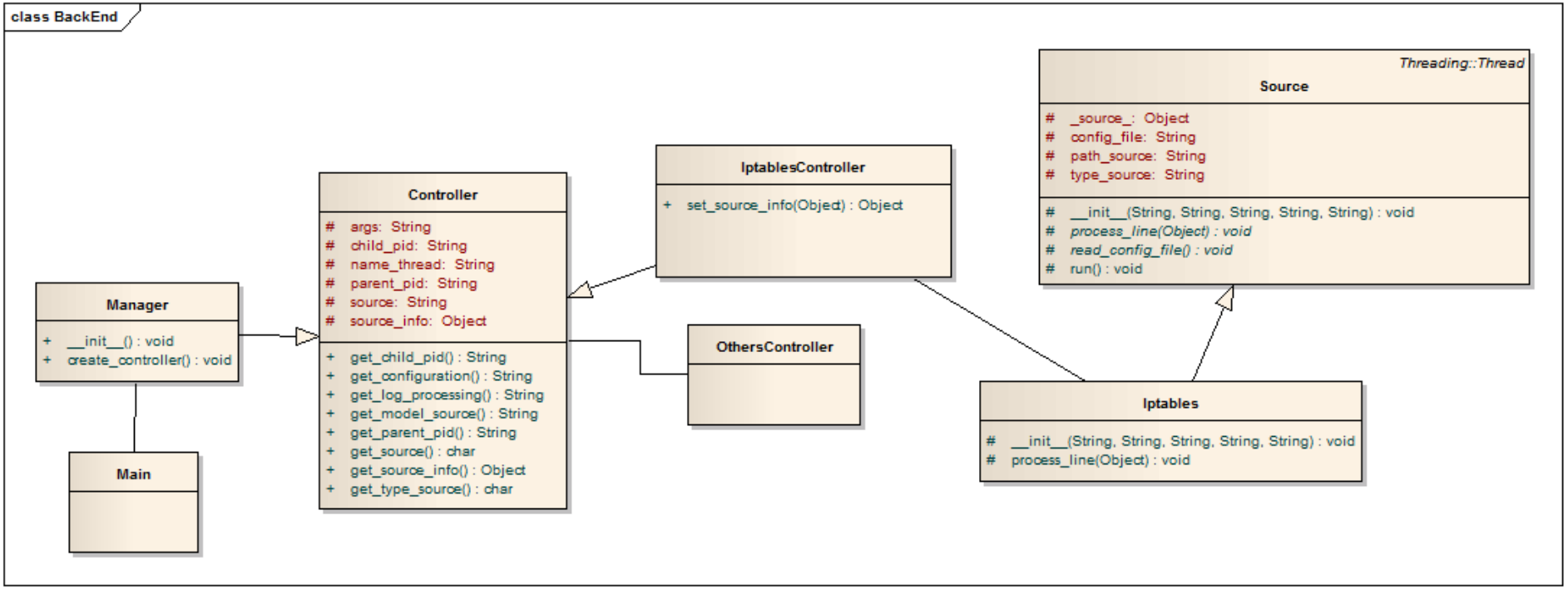


Figura - Diagrama de clases Backend

Clases del sistema

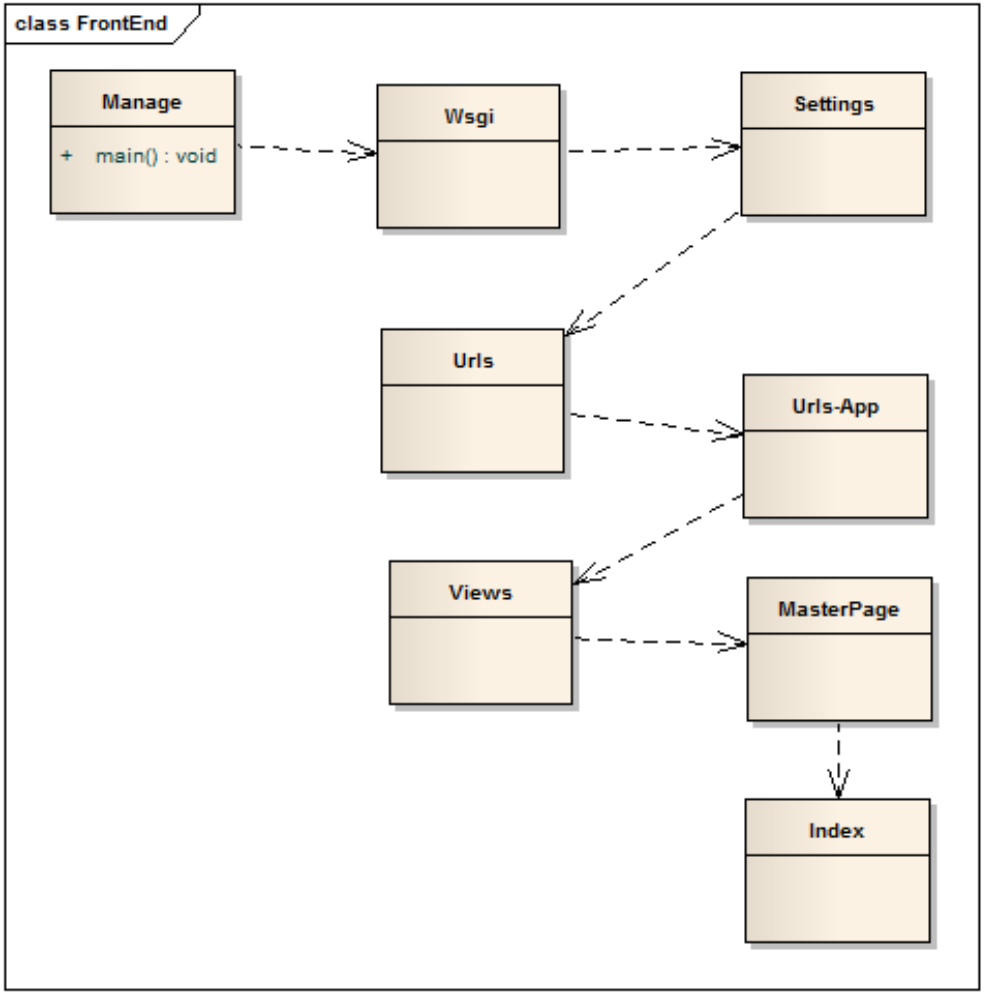
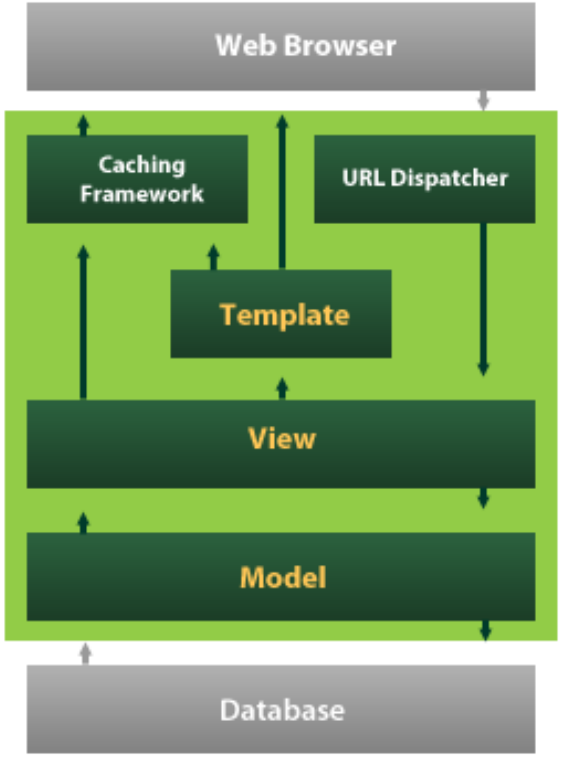


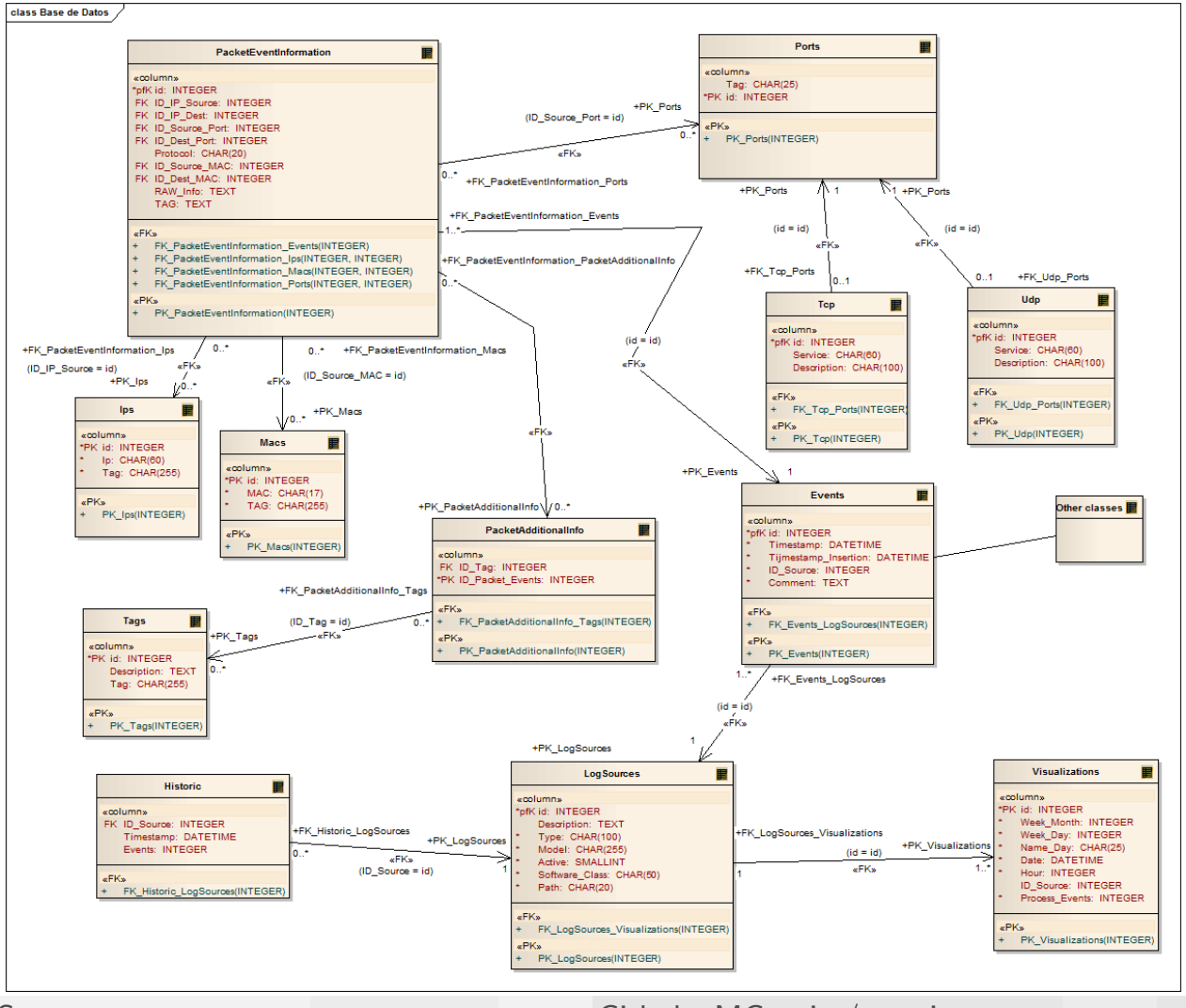
Figura - Diagrama de clases Frontend

Workflow Django



http://mytardis.readthedocs.io/en/3.5/_images/DjangoArchitecture-JeffCroft.png

Base de datos



Rsyslog

```
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 - Configuración de iptables para Rsyslog

Logrotate

```
1     /var/log/iptables.log
2     {
3       rotate 7
4       daily
5       missingok
6       notifempty
7       delaycompress
8       compress
9       postrotate
10      invoke-rc.d rsyslog restart > /dev/null
11      endscript
12    }
```

Figura - Configuración de iptables para Logrotate

Iptables

```
1  # Generated by iptables-save v1.4.21 on Mon Jan 25 20:37:18 2016
2  *filter
3  :INPUT ACCEPT [0:0]
4  :FORWARD ACCEPT [0:0]
5  :OUTPUT ACCEPT [0:0]
6  -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 "
7  -A INPUT -d 127.0.0.1/32 -p icmp -m icmp --icmp-type 8 -m state --state
       NEW,RELATED,ESTABLISHED -j DROP
8  -A INPUT -p tcp -m tcp --dport 22 -j LOG --log-prefix "IPTMSG=Connection
       SSH "
9  -A INPUT -p tcp -m tcp --dport 22 -j DROP
10 COMMIT
```

Figura - Configuración reglas iptables

Eventos de Iptables

```
1 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
2 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 - Log capturado y almacenado por rsyslog en /var/log/iptables.log

Correlación

```
1 obj = Pygtail("/var/log/iptables.log")
2
3 while True:
4     try:
5         for line in obj:
6             if len(line) > 1:
7                 self.process_line(line)
8     except Exception as ex:
9         print "Pygtail processing -> ", ex
```

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

Main

```
1
2  import signal
3  import os
4  import manager
5  import re
6
7
8  if __name__ == "__main__":
9
10     # Instanciamos la clase Manager para la ejecucion de las distintas fuentes
11     manager_obj = manager.Manager()
12
13     # Obtenemos la instancia de la fuente iptables para su ejecucion devuelta
           por el controlador asociado
14     iptables = manager_obj.create_controller('iptables')
15
16     # Si establecemos iptables como un pid independiente tendremos que hacer
           uso de fork para esto y
17     # asi el hilo Main sera sobre el cual se haran los operaciones
18
19     if not iptables.get_child_pid() == iptables.get_parent_pid():
20         threads = {'Main thread': iptables.get_parent_pid(), 'Thread-1':
               iptables.get_child_pid()}
21     else:
22         threads = {'Main thread': iptables.get_parent_pid()}
```

Test de integridad - Events

```
1  class EventsTestCase(TestCase):
2
3      # Atributos internos de la clase
4
5      timestamp = timezone.now()
6      timestamp_insertion = timezone.now()
7
8      def setUp(self):
9          log_sources = LogSources.objects.create(
10             Description="Firewall of gnu/linux kernel",
11             Type="Iptables",
12             Model="iptables v1.4.21",
13             Active=1,
14             Software_Class="Firewall",
15             Path="iptables",
16         )
17         Events.objects.create(
18             Timestamp=self.timestamp,
19             Timestamp_Insertion=self.timestamp_insertion,
20             ID_Source=log_sources,
21             Comment="Iptables Events",
22         )
```

Figura - Test unitario clases del modelo - Events

Test de integridad - Events

```
1 def test_events_timestamp(self):
2     """
3     Comprobacion de que el timestamp del evento corresponde al del asociado
4     Returns:
5
6     """
7     events = Events.objects.get(Timestamp=self.timestamp)
8     self.assertEqual(events.get_timestamp(), self.timestamp)
```

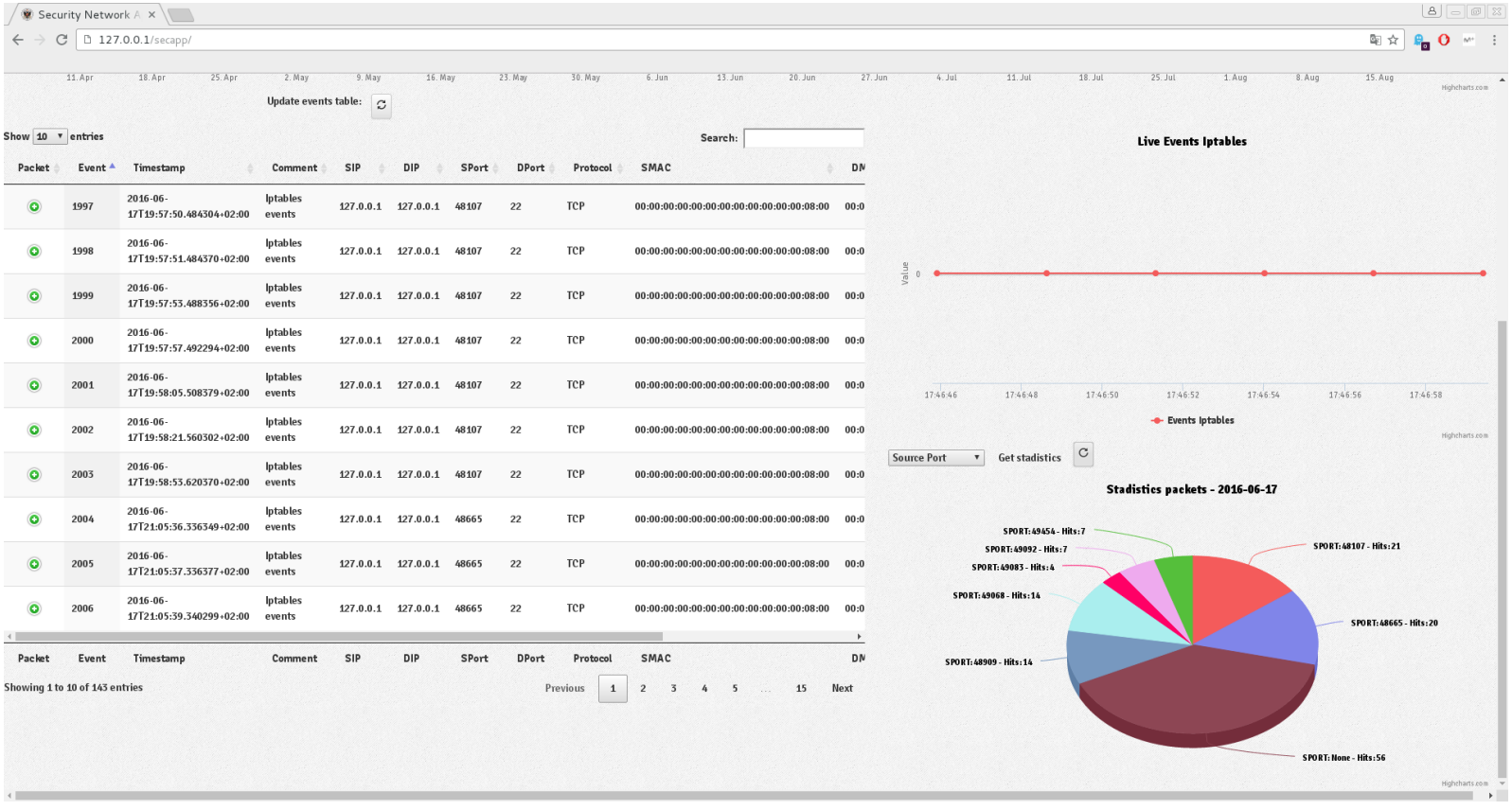
Figura - Método de la clase Test test_events_timestamp

Conclusión

Durante la realización de éste proyecto, se han conseguido los siguientes resultados:

- Comprender el funcionamiento de los servicios del sistema: rsyslog, syslog, logrotate, nginx.
- Comprender el funcionamiento del firewall del kernel, Iptables.
- Desarrollar una solución software usando el lenguaje de programación Python.
- Desarrollar una solución software, para sistemas web, usando el framework Django.
- Manipulación de bases de datos relacionales usando un modelo orientado a objetos (ORM) proporcionado por el framework de desarrollo.
- Diseñar el prototipo base para los sensores de recopilación de información de seguridad.
- Diseñar una interfaz web donde el usuario pueda visualizar la información que el sensor ha analizado.
- Despliegue de la aplicación en un servidor web en la nube para realizar demostraciones de la herramienta (Digital Ocean).
- Documentar todo el proceso de realización del proyecto.

Demostración



Final

¿Preguntas?

Figura - http://ph03nyx.deviantart.com/
Para más información: https://github.com/MGautier/security-sensor