# HTTPSpy

Marasca, Rodríguez, Di Pietro, Scally

FCEN, UBA

5 de julio de 2012

# Objetivos iniciales

## Objetivo inicial

Se deberá implementar un sniffer http que permita escuchar el tráfico hacia un proxy http, para luego facilitar el análisis del tráfico capturado.

Características:

- El tráfico deberá almacenarse en una base de datos SQL.
- Proveerá una interfaz para realizar consultas sobre la base de datos.
- Se almacenará ip-origen, fecha y hora, método http, URL.
- De ser posible se almacenarán los headers de la respuesta, incluyendo código de respuesta, tamaño de la misma y Content-Type.
- Se podrán generar reportes de anomalías.

# Objetivos adicionales

Objetivos adicionales:

- Se podrá filtrar el tráfico a almacenar según reglas ingresadas por el usuario (hosts, puertos, etc.).
- El usuario podrá seleccionar que parte de la conversación HTTP desea almacenar (hosts, puertos, método, headers, etc.).
- Podrán utilizarse diferentes formas de almacenar el tráfico sniffeado (SQL, texto plano, XML, YAML, etc.).
- La herramienta deberá ser robusta y no romperse por anomalías en el tráfico.
- Será deseable que se trate de utilizar sólo herramientas/módulos estándares del lenguaje seleccionado para su implementación.
- Deberá ser un sistema muy simple y poco acoplado, de forma de permitir que se utilice para implementar sistemas más complejos.
- La implementación será compacta y portable.

# Request HTTP

```
GET / HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0) Opera 7.11
Host: 10.1.1.1
Accept: text/xml,application/xml,application/xhtml+xml,text/html
Accept-Language: en
Accept-Charset: windows-1252, utf-8, utf-16, iso-8859-1;q=0.6, *;q=0.1
Accept-Encoding: deflate, gzip, x-gzip, identity, *;q=0
Connection: Keep-Alive
```

# Response HTTP

```
HTTP/1.1 200 OK
Date: Sat, 20 Nov 2004 10:21:06 GMT
Server: Apache/2.0.40 (Red Hat Linux)
Last-Modified: Mon, 08 Mar 2004 20:27:54 GMT
ETag: "46eed-a0-800ce680"
Accept-Ranges: bytes
Content-Length: 160
Connection: close
Content-Type: text/html; charset=ISO-8859-1

<html>
<head>
<title>Y la triple corona?</title>
</head>
<body>
...
</body>
</html>
```

# Request HTTPS

```
CONNECT www.google.com:443 HTTP/1.0
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:13.0) Firefox/13.0.1
Host: www.google.com
Content-Length: 0
Proxy-Connection: Keep-Alive
```

# Response HTTPS

```
.........|ia..R..Z*..j.$....Z.X$g/D46.\.]...j
J.[.C...x*=..%...%^.8m.....%c..  ..5...~./.]..z...e.}...|
.v.......+.[.v....#qC.x.d....GK...B‘.KM1)]...21...;..
T%.......(.......).d.......U........V...W.u..dB...Lz..!.?...x
...9....S.......nI.y..;.v.,..P...
S..xl^^T...._~"Y..5..V...X".%........a....C?.....-..<...
.n..e... h.1.f.0-?.....a..|Ygy8H.u.l-d...S...<.......z.c.
.....J...1>..L.....6u..b&...QV|1.5.q....x?Y..E.6..
........1.+.....^.h......4."..H..i...Pk...
.F6..._xR.X.j....E.A...2<’b...N$.[.<Q;^w......_K.]..,.
./.j.@........’...rC...w}..  ....9.D..ii+.:w;.....
......’..w.c3g.......f.....
..i..N.L......vjE.....M.~..B.Al
..........q.V.yQ3/J...
```

# Herramientas analizadas

Se analizaron las siguientes tecnologias para la implementación del proyecto.

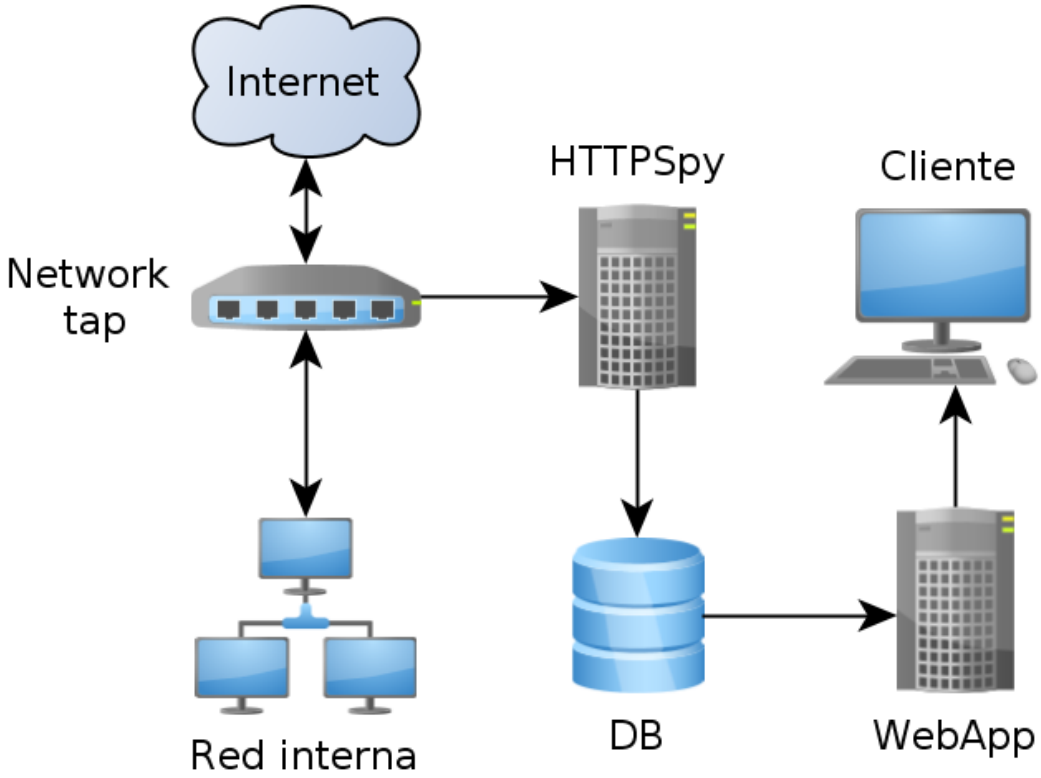
- Lenguajes: Python, C/C++, Java, Perl.
- libpcap, pycap, scapy.
- libnids, pynids.
- http_parser.
- MySql, SQLite, PostgreSql, Oracle, Sql Server, SQLAlchemy.
- Mojolicious, Djando, Bottle, Web2py.
- ncurses.
- tinyproxy.

# Herramientas utilizadas

Para el desarrollo utilizamos las siguientes tecnologías.

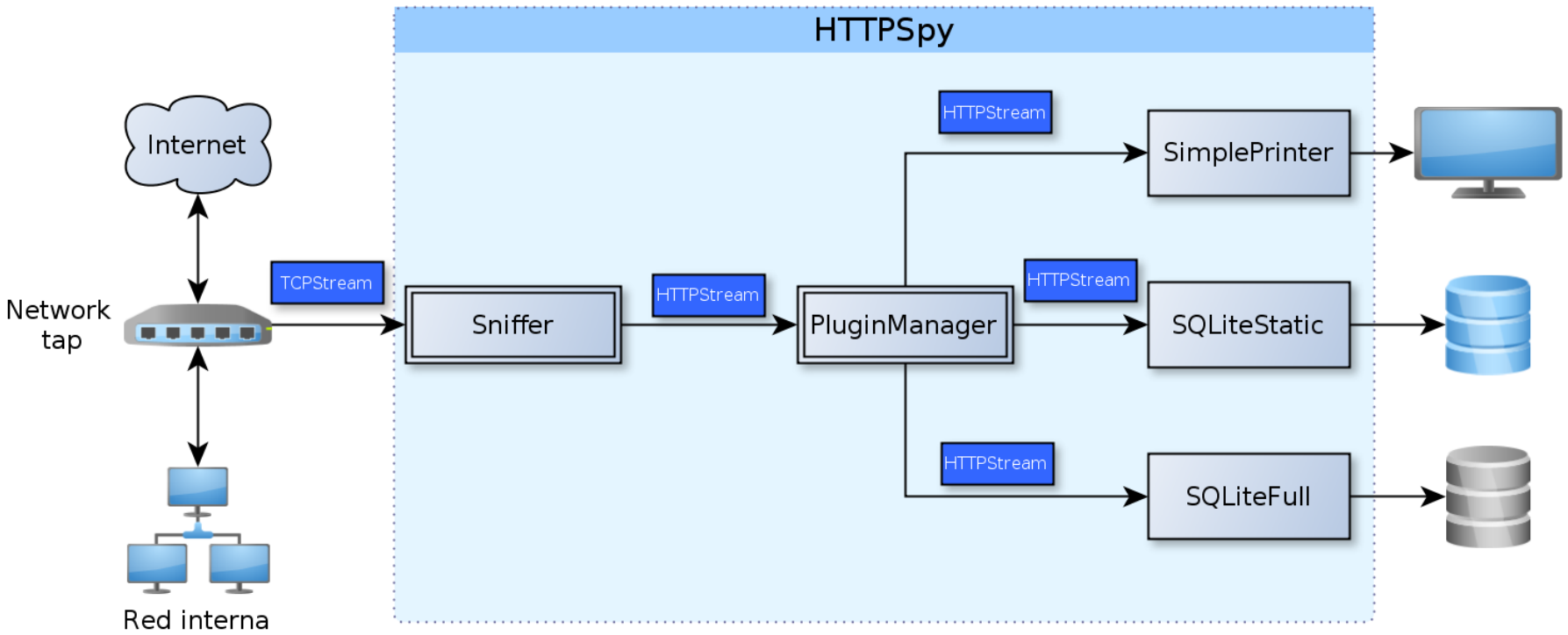
- Lenguajes: Python
- Sniffing: pynids.
- Parser: http_parser.
- Almacenamiento: SQLite.
- Framework Web: Mojolicious.
- Auxiliares: tinyproxy, tcpdump, wireshark, etc.

# Deploy




Ejemplo de deploy

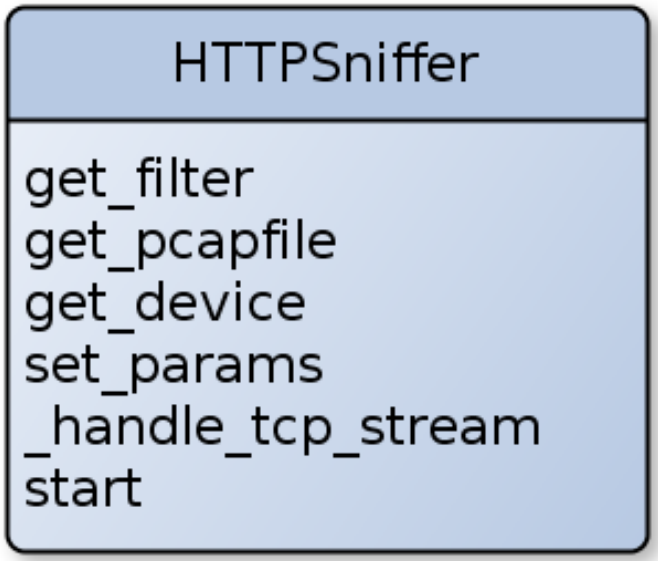
# Diseño de la aplicación



# HTTPSniffer

## Responsabilidad

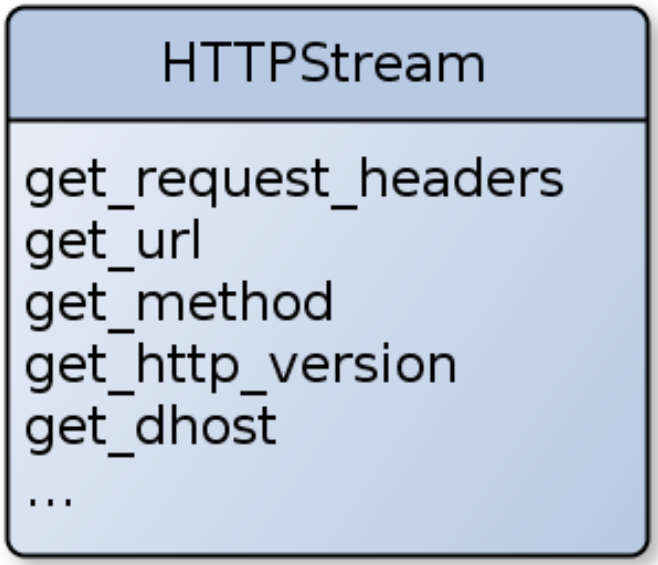
Encargado de capturar el tráfico según las reglas de filtrado definidas. Por cada conversación HTTP genera una instacia de HTTPStream y ejecuta la función de callback pasándolo como parámetro. Es un singleton que encapsula a la librería pynids.



# HTTPStream

## Responsabilidad

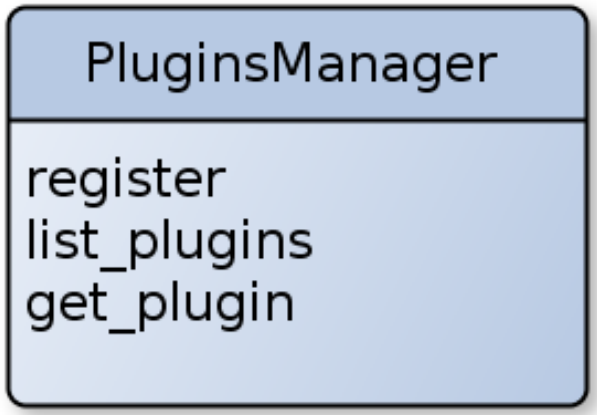
Representa una conversacion HTTP, almacena toda la información interesante sobre la misma. No alacena el payload de la conversación.



# PluginsManager

## Responsabilidad

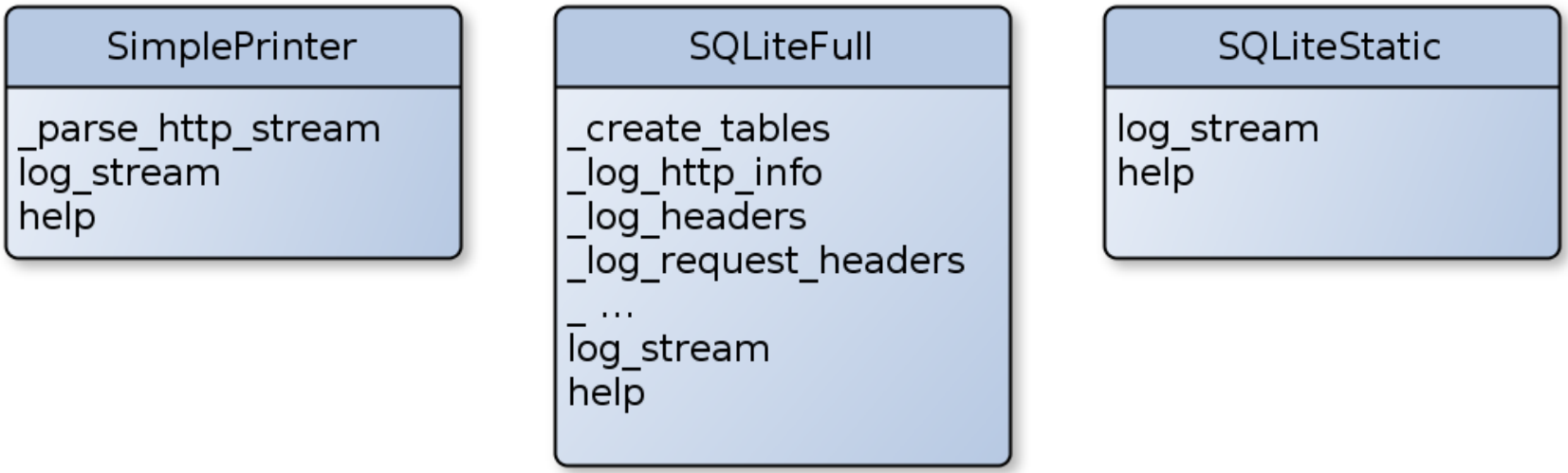
Administra los plugins instalados en la aplicación. Asocia el nombre del plugin con la clase correspondiente.



# Plugins

## Responsabilidad

Son los encargados de procesar los requests capturados por el sniffer. Cada uno debe implementar los métodos log_stream y help.



# Plugins implementados

## SimplePrinter

Este plugin imprime por stardard output, según un formato configurable, información sobre la conversacion HTTP.

## SQLiteStatic

Almacena en una unica tabla SQLite los siguientes campos: source host, destination host, request path, request method, status code, content length, content type.

## SQLiteFull

Almacena en tres tablas SQLite toda la información relacionada a la conversación, incluyendo todos los headers.

# Instalación Debian likes

## Repositorio tentativo

`https://github.com/cdipietro/2012_1c_seginfo`

Instalación paso a paso:

1. apt-get install python python-pip python-dev python-nids
2. pip install http_parser
3. cpan App::cpanminus
4. cpanm DBI DBD::SQLite Mojolicious
5. Enjoy!

# Modo de uso

```
usage: main.py [-h] [--pcapfile PCAPFILE | --device DEVICE]
[--filter FILTER]
[--list-plugins | --help-plugin HELP_PLUGIN | --plugins PLUGINS]

A very basic http sniffer

optional arguments:
  -h, --help            show this help message and exit
  --pcapfile PCAPFILE   read a tcp stream from a file
  --device DEVICE       set the device to sniff
  --filter FILTER       set the filter (see man tcpdump)
  --list-plugins        List availables plugins
  --help-plugin HELP_PLUGIN
                        Show help for a plugin
  --plugins PLUGINS     File with configured plugins
```

# Ejemplo de uso

printer.yml:

```
- name      : SimplePrinter
  delimiter : "\t"
  format    : "shost method >Host path status_code <Content-Type"
```

Comando para capturar de un dispositivo:

```
$python main.py --device eth0 --plugins printer.yml
```

Comando para procesar un pcapfile:

```
$python main.py --pcapfile trafico.pcap --plugins printer.yml
```

Salida:

```
192.168.0.12 GET www.gnu.org / 200 text/html
```

# Demo de plugins


Demo

# Interfaz Web

Es una pequeña aplicación web que permite realizar consultas predefinidas sobre la base de datos generada por el sniffer.

1. Los informes preconfigurados se definen mediante un archivo de configuración con formato yaml.
2. Se pueden definir informes parametrizados.
3. Aún se encuentra en fase de desarrollo.

Ejemplo:

```
- nombre: Listar fecha, origen, destino (like)
  query: 'SELECT date, shost, host FROM http_log WHERE host LIKE ?'
  campos:
   - Destino
  columnas:
   - Fecha
   - Origen
   - Destino
```

# Demo de la Interfaz Web


Demo

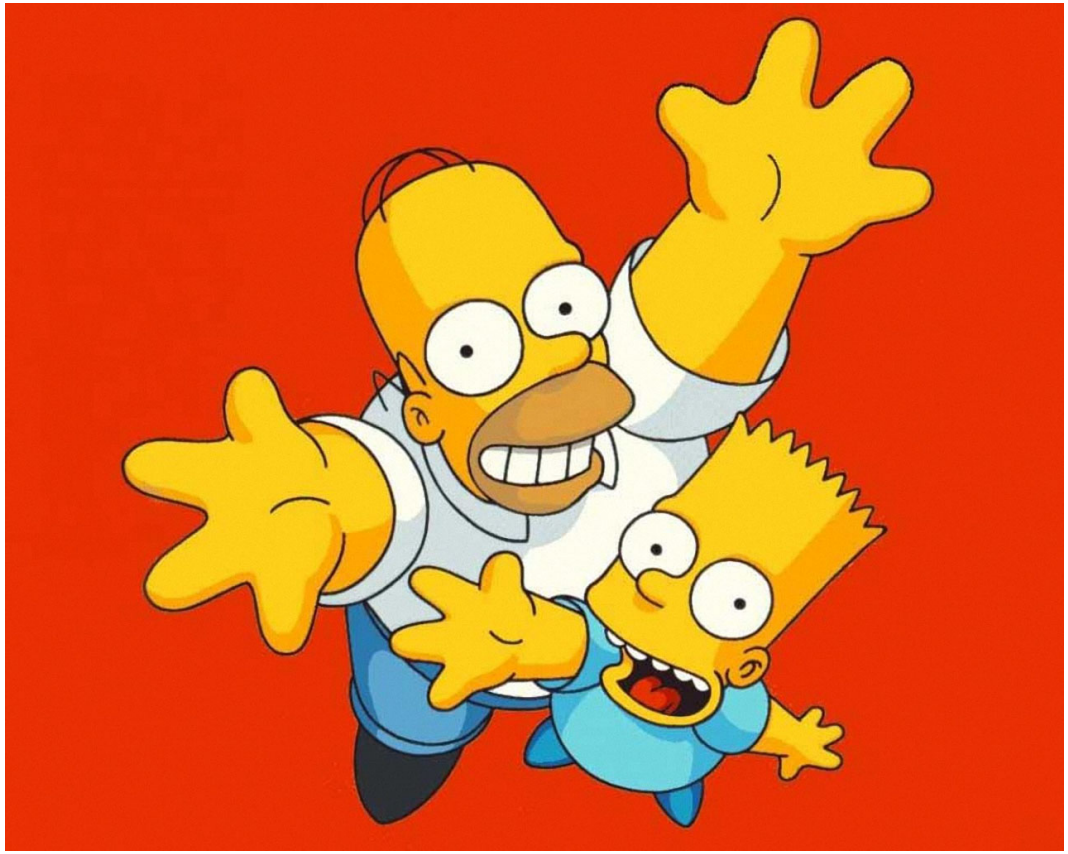
# Mejoras a Futuro

- Mejorar la robustez de la Interfaz web.
- Programar una interfaz basada en ncurses.
- Programar más tests de unidad.
- Pruebas de performance en ambientes productivos de gran escala.
- Plugin de Syslog.
- Plugin Oracle, PostgreSQL, Mysql y SQLServer o SQLAlchemy.
- Script de instalación.
- Mejorar la documentación.
- Reestruturar el código para subirlo a "Python Package Index".
- Crear un repositorio y una web para la aplicación (wiki + bugtracker).

# Preguntas

¿Preguntas?

# Fin



Gracias