DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico

05 de Agosto de 2012 — Seguridad de la Información

Grupo "Sniffer de Proxy HTTP/HTTPS"

Integrante	LU	Correo electrónico
Di Pietro, Carlos Augusto Lyon	126/08	cdipietro@dc.uba.ar
Marasca, Dardo Gustavo	227/07	gefarion@gmail.com
Rodríguez, Alejandra Alicia	077/04	alerodriguez4989@gmail.com
Scally, Tomás Santiago	886/03	horizontedesucesos@gmail.com

Facultad de Ciencias Exactas y Naturales

Universidad de Buenos Aires

Ciudad Universitaria - (Pabellón I/Planta Baja)

Intendente Güiraldes 2160 - C1428EGA

Ciudad Autónoma de Buenos Aires - Rep. Argentina

Tel/Fax: (54 11) 4576-3359

http://exactas.uba.ar

Índice

1. Introducción

1.1. Objetivos

1.1.1. Objetivos Iniciales

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.
- Deberá permitir realizar diferentes informes sobre los datos almacenados.

1.1.2. Objetivos adicionales

Características añadidas:

- 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 (SQLite, Syslog, MySQL, texto plano por consola, 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.
- Se podrá notificar al usuario, vía mail, de accesos considerados críticos por el usuario (i.e., acesos que cumplan ciertas condicones informadas por el usuario).

1.2. Breve Introducción Teórica

1.2.1. Protocolos HTTP y HTTPS

HTTP (HyperText Transfer Protocol) es el protocolo más común de intercambio de información en la WWW (World Wide Web). HTTP define la sintaxis y la semántica que utilizan los elementos de software de la arquitectura web (clientes, servidores, proxies) para comunicarse. Es un protocolo orientado a transacciones y sigue el esquema de petición-respuesta entre un cliente y un servidor. Una transacción HTTP está formada por un encabezado seguido, opcionalmente, por una línea en blanco y algún dato. El encabezado especificará cosas como la acción requerida del servidor, o el tipo de dato retornado, o el código de estado.

HTTPS (Hypertext Transfer Protocol Secure) es una combinación del protocolo HTTP y protocolos criptográficos. Se emplea para lograr conexiones más seguras en la WWW, generalmente para transacciones de pagos o cada vez que se intercambia información sensible (por ejemplo, claves) en Internet. De esta manera la información sensible, en el caso de ser interceptada por un intruso, estará cifrada.

El sniffer que implementamos captura tráfico HTTP y HTTPS. A modo de ejemplo les mostramos a continuación la forma de los headers de ambos métodos.

El que prosigue a continuación es un ejemplo de Request HTTP:

```
GET / HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0) Opera 7.11  [en]
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
```

El cual invoca como respuesta un Response HTTP como el siguiente:

```
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>
```

Por su parte tenemos un 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
```

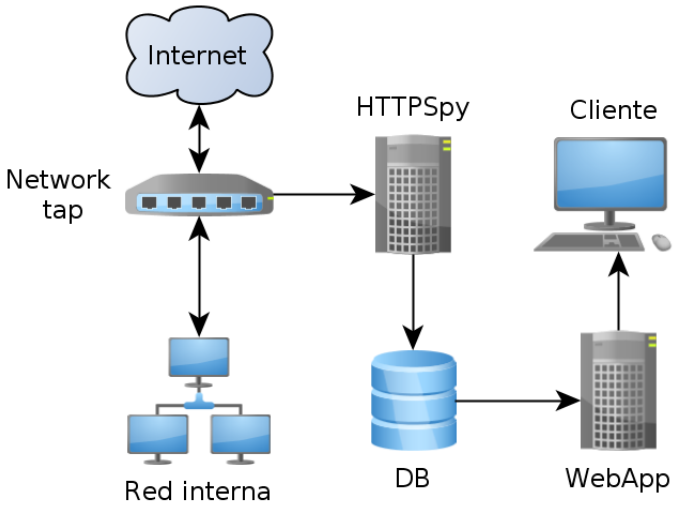
Y su respectivo response HTTPS, que como puede notarse llega cifrado:

```
.........|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...
```

1.2.2. Sniffer

Un “Sniffer” o “Analizador de Paquetes” es un programa o aparato de monitorea los datos que viajan sobre una red. Pueden ser usados tanto para detectar intrusos o problemas, como así también por atacantes para robar información de una red. Su función es capturar los paquetes o tramas que circulan en la red para luego su posterior análisis. Los paquetes son cada uno de los bloques en que se divide la información que se envía a través de una red en el nivel de red del modelo OSI. Por debajo de este nivel el paquete adquiere el nombre de trama de red.

El sniffer que implentamos monitorea los paquetes de la red y captura los pertenecientes a los protocolos HTTP y HTTPS. A modo de ejemplo podemos ver en el siguiente gráfico como estaría situado nuestro sniffer:



Todo el tráfico que pasa entre la red interna e Internet debe pasar por el Network Tap (en nuestro caso un proxy) y allí es cuando ese tráfico es capturado por el sniffer que implementamos (HTTPSpy). El mismo si detecta paquetes pertenecientes al protocolo HTTP o HTTPS rearma dichos paquetes y los almacena en una base de datos para su posterior análisis. Finalmente, un usuario cliente puede ingresar a la aplicación web y realizar consultas sobre datos a partir de los registros capturados y almacenados en la base de datos por el sniffer. Adicionalmente, aunque no está grafico, nuestro sniffer también puede conectarse con un servidor smtp y enviar un mail al cumplirse ciertas condiciones sobre las conexiones.

2. Desarrollo

2.1. Herramientas utilizadas

Para la implementación del sniffer 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.

De esas herramientas nos decidimos en lenguaje por uno de bajo nivel. Y nos inclinamos por Python debido a que gran parte del grupo quería aprender a programar en dicho lenguaje o mejorar su experiencia de programación en el mismo. Su sintaxis además es simple y clara; dispone de tipado dinámico, de gran cantidad de librerías disponibles y es un lenguaje potente. Finalmente también otra punto a su favor es que es un lenguaje bastante usado en el ámbito del desarrollo ligado a la seguridad.

En cuanto a las librerías de python relacionadas a la captura de tráfico que consideramos para realizar la implementación:

- PYCAP: es un módulo de bajo nivel que permite capturar e inyectar paquetes en cualquier interfaz de red, esta basada en libpcap y libnet. El principal problema que presenta es que solamente implementa hasta la capa TCP y no permite rearmar el flujo de la conversación TCP entre los hosts.
- SCAPY: basada en PYCAP, es una implementación más modular que la anterior que permite definir de forma muy fácil nuevas capas. Si bien pudimos definir una capa para el tráfico HTTP, al igual que PYCAP, no permite de forma nativa rearmar los streams TCP.
- PYNIDS: es una librería en python basada en libnids. A diferencia de las anteriores, PYNIDS emula el stack IP del kernel de Linux, ofrece defragmentación IP, ensamblado del stream TCP y detección de scaneo de puertos. Todo esto nos llevo a seleccionar esta librería.

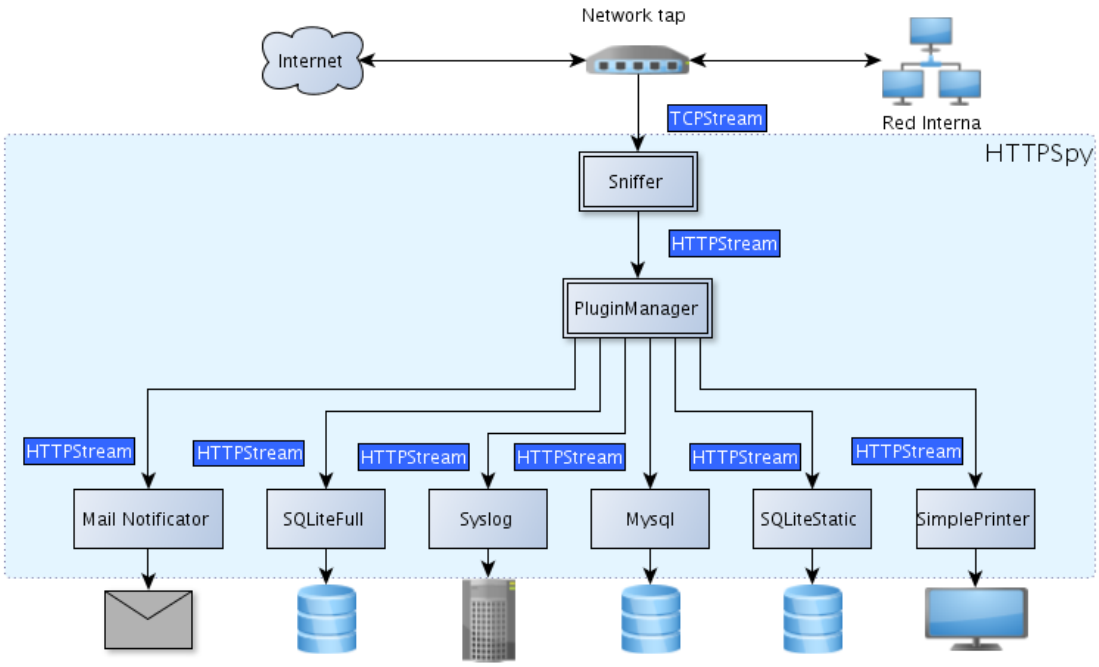
Las 3 herramientas poseen la posibilidad de sniffear múltiples interfaces utilizando diferentes filtros para el tráfico.
Se implementó un pequeño sniffer http con las 3 herramientas mencionadas. Si bien con todas ellas es bastante simple capturar tráfico, la mayor dificultad la encontramos al intentar rearmar el stream TCP, la única que puede realizar esto de forma nativa es PYNIDS. Por lo tanto en caso de querer utilizar PYCAP o SCAPY para nuestra

implementación, deberíamos desarrollar por nuestra cuenta todo el ensamble y las conversaciones TCP entre los hosts. Por este último motivo elegimos PYNIDS.
Para interpretar y parsear la conversación HTTP se utilizarla librería http-parser, la cual soporta de forma completa la especificación de dicho protocolo.
Para almacenar el tráfico capturado utilizamos SQLite, dado que esto nos permite alcanzar el objetivo de portabilidad que anteriormente mencionamos. Decidimos que no valía la pena utilizar un motor de base de datos más complejo como primer implementación, pero si tiene que estar la posibilidad de utilizar uno en un futuro.
Y, finalmente, para implementar la aplicación web, usada para consultar los datos capturados por el sniffer, nos decantamos por el framework web Mojolicious debido a la experiencia con dicha herramienta por parte de uno de los miembros del grupo.

Finalmente, se aclara que todo el trabajo fue desarrollado sobre Debian Linux.

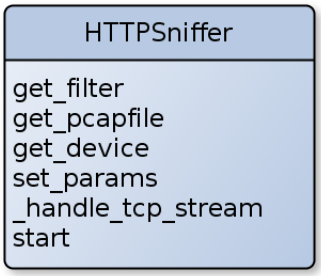
2.2. Diseño de la Aplicación

Buscando alcanzar las características deseadas mencionadas anteriormente, decidimos dividir el sistema en tres grandes partes como se puede ver en el siguiente esquema (Sniffer, PluginManager y los plugins en sí):



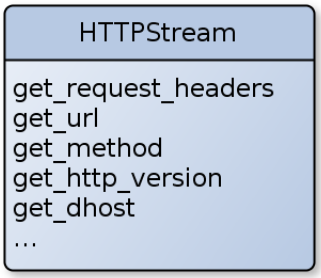
2.2.1. HTTPSniffer

Es el sistema de captura, es el encargado de capturar y extraer de las conversaciones HTTP/S la información relevante 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.



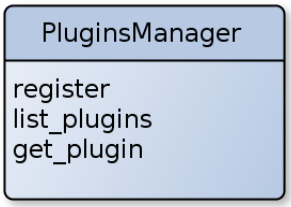
2.2.2. HTTPStream

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



2.2.3. PluginsManager

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



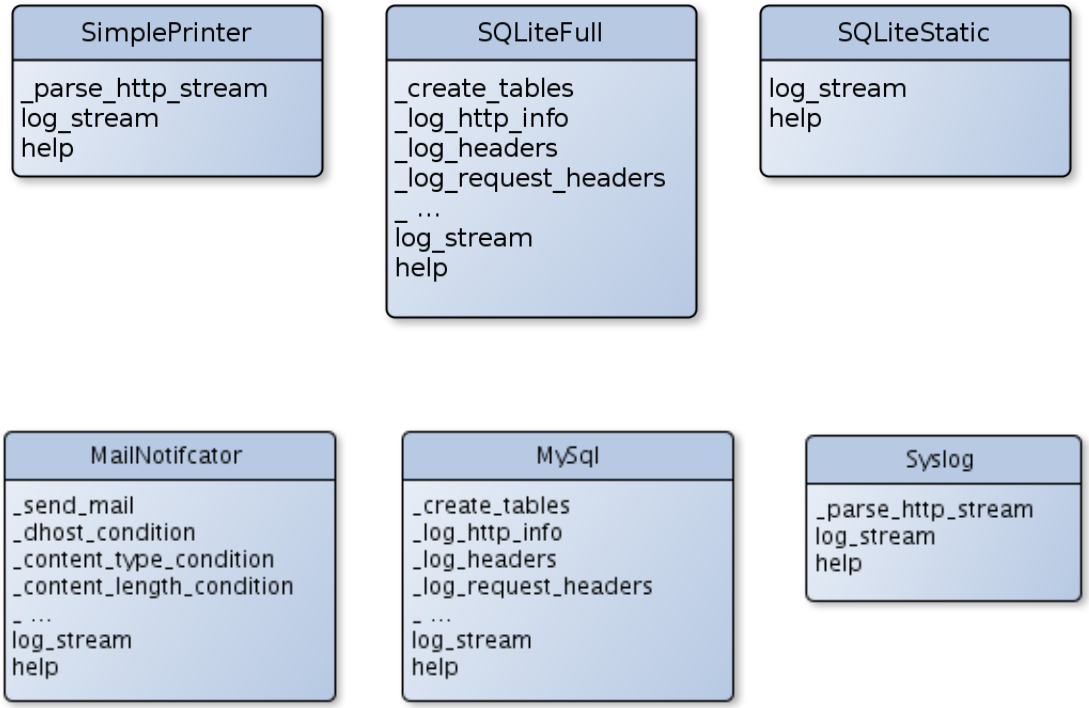
2.2.4. Plugins

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.
- **SyslogPrinter:** Este plugin registra por Syslog, 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.
- **MySQLFull:** Almacena en tres tablas MySQL toda la información relacionada a la conversación, incluyendo todos los headers.
- **MailNotificator:** Envía por mail notificaciones de conexiones que cumplen ciertas condiciones críticas informadas. Envía datos tales como source host, destination host, source port, destination port, request url, request path, request method, status code, content type, content type, time of the request.



3. Utilizando HTTPSpy

3.1. Instalación Debian likes

Desde una consola con privilegios de administrador, ejetutar los siguientes pasos para instalar las primero las dependencias y luego la aplicación.

- Instalación de dependencias:
  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
- Instalación de la aplicación:
  1. wget http://http-spy.googlecode.com/files/HTTPSpy-1.0.tar.gz
  2. tar xvzf HTTPSpy-1.0.tar.gz
  3. cd HTTPSpy-1.0
  4. python setup.py install

3.2. Modo de Uso

HttpSpy viene con una opción de help que indica como usar el mismo. Ingresando "httspy.py -h"se obtienen las siguientes intrucciones de uso:

```
usage: httpspy.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:

```
Archivo de ejemplo de configuración de un módulo, printer.yml:
- name      : SimplePrinter
  delimiter : "\t"
  format    : "shost method >Host path status_code <Content-Type"

Comando para capturar de un dispositivo:
$httpspy.py --device eth0 --plugins printer.yml

Comando para procesar un pcapfile:
$httpspy.py --pcapfile trafico.pcap --plugins printer.yml

Ejemplo de salida:
192.168.0.12 GET www.gnu.org / 200 text/html
```

3.3. Interfaz Web

3.3.1. Instalando la Aplicación Web

Pasos de la instalación:

```
> sudo cpan cpan App::cpanminus
> sudo cpanm DBI
> sudo cpanm DBD::SQLite
> sudo cpanm YAML::Tiny
> sudo cpanm Mojolicious
```

3.3.2. Utilizando la Aplicación Web

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

- Los informes preconfigurados se definen mediante un archivo de configuración con formato yaml.
- Se pueden definir informes parametrizados.

Para ponerla en ejecución hay que ejecutar la siguient sentencia desde el directorio de la web app.

```
> morbo informes
[Sat Aug  4 23:43:38 2012] [info] Listening at "http://*:3000".
Server available at http://127.0.0.1:3000.
> _
```

Una vez puesta en ejecución podremos acceder a la misma desde la url informada en el resultado del comando y realizar desde ella consultas sobre los datos almacenados de los accesos http/https. Además en http://localhost:3000/query se pueden hacer consultas sql en caso de que no este predefinida la busqueda deseada. Para el ingreso de sitios en la lista negra de la aplicación se provee un formulario en http://localhost:3000/lista_negra.

Por ejemplo una de las consultas definidas es la siguiente:

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

4. Conclusiones

4.1. A modo de nota final

A modo de conclusión podemos citar que al realizar el programa de manera modular nos llevo a que pudiéramos agregar más funciones de manera sencilla.
Las librerías utilizadas para el análisis del tráfico también nos ayudaron a analizar de manera más eficiente el tráfico http/https de la red. Pudiéndo así centrarnos más en que hacer con los datos obtenidos que en como obtenerlos.
El uso de pcapfiles para algunas de las pruebas nos ayudo a probar más rápidamente el funcionamiento correcto de los diferentes módulos implementados para el almacenamiento e impresión de datos, como así también para probar las consultas a la base de datos desde la interfaz web.

5. Fuentes

- http://pycap.sourceforge.net/
- http://www.secdev.org/projects/scapy/
- http://jon.oberheide.org/pynids/
- http://www.freesoft.org/CIE/RFC/1945/index.htm