

# Sistema Proxy-Web con Filtrado y Control de acceso

**Ignacio Hoces-Moral López**

Grado de Ingeniería Informática

Administración de redes y sistemas operativos

**Manuel Jesus Mendoza Flores**

**Javier Panadero Martínez**

09/06/2019



Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada [3.0 España de Creative Commons](https://creativecommons.org/licenses/by-nc-nd/3.0/es/)

## FICHA DEL TRABAJO FINAL

<b>Título del trabajo:</b>	<i>Sistema Proxy-Web con Filtrado y Control de acceso</i>
<b>Nombre del autor:</b>	<i>Ignacio Hoces-Moral López</i>
<b>Nombre del consultor/a:</b>	<i>Manuel Jesus Mendoza Flores</i>
<b>Nombre del PRA:</b>	<i>Javier Panadero Martínez</i>
<b>Fecha de entrega (mm/aaaa):</b>	06/2019
<b>Titulación::</b>	<i>Grado de Ingeniería Informática</i>
<b>Área del Trabajo Final:</b>	<i>Administración de redes y sistemas operativos</i>
<b>Idioma del trabajo:</b>	<i>Castellano</i>
<b>Palabras clave</b>	<i>Proxy, Seguridad en navegación, Inspección del SSL</i>

**Resumen del Trabajo (máximo 250 palabras):** *Con la finalidad, contexto de aplicación, metodología, resultados y conclusiones del trabajo.*

El ancho de banda se ha convertido en un coste de infraestructura muy importante en los últimos años. El tamaño de los contenidos ha crecido de forma exponencial gracias a mayor capacidad de las comunicaciones.

Las organizaciones tienen la necesidad de saber para qué se utiliza este recurso. El acceso a las redes es uno de los principales vectores de ataque tanto de virus como de *malware* o *ransomware*. Por otro lado, es el método más utilizado para la fuga de información o para la pérdida de rendimiento de los usuarios.

Para alcanzar ese objetivo, este TFG plantea un sistema potente, flexible, escalable y de bajo coste. Este sistema está formado por varias herramientas interrelacionadas, que cumplen cada una de ellas una función específica orientada a controlar o informar del consumo de ancho de banda de internet.

Los principales componentes de este sistema son el servidor proxy-cache, encargado de ser el intermediario entre las peticiones del cliente e internet.

Otro de los componentes es el motor de antivirus de navegación que asegurará que los contenidos entregados no representan una amenaza para la organización.

El siguiente componente es el sistema de balanceo de carga. Este servicio dotará de escalabilidad al sistema, haciendo que pueda implantarse en

distintos tipos de organización según varíen sus necesidades.

El último componente es el recopilador y analizador de logs, el cual, dotará al sistema de las herramientas necesarias para transformar los datos en conocimiento para la organización.

**Abstract (in English, 250 words or less):**

Bandwidth has become a very important infrastructure cost in recent years. The size of the content has grown exponentially thanks to greater capacity of communications.

Organizations have the need to know what this resource is used for. Access to networks is one of the main attack vectors of both viruses and malware or ransomware. On the other hand, it is the most used method for the leakage of information or for the loss of user performance.

To achieve this goal, this TFG proposes a powerful, flexible, scalable and low-cost system. This system is made up of several interrelated tools, each of which fulfills a specific function oriented to control or inform the consumption of internet bandwidth.

The main components of this system are the proxy-cache server, in charge of being the intermediary between the client's requests and the Internet.

Another component is the navigation antivirus engine that will ensure that the content delivered does not represent a threat to the organization.

The next component is the load balancing system. This service will provide scalability to the system, allowing it to be implemented in different types of organizations as their needs vary.

The last component is the log collector and analyzer, which will provide the system with the necessary tools to transform the data into knowledge for the organization.

# Índice

## Sumario

1. Introducción.....	1
1.1 Contexto y justificación del Trabajo.....	1
1.2 Objetivos del Trabajo.....	2
1.3 Enfoque y método seguido.....	3
1.4 Planificación del Trabajo.....	3
1.5 Breve sumario de productos obtenidos.....	4
1.6 Breve descripción de los otros capítulos de la memoria.....	4
2. Diseño y elección de componentes del sistema.....	5
2.1 Diseño del sistema.....	5
2.2 Servicio de proxy-web.....	6
2.3 Servicio de balanceo de carga.....	10
2.4 Servicio de descifrado SSL.....	11
2.5 Servicio de Antivirus de Navegación.....	13
2.6 Centralización de logs.....	14
3. Implementación de la solución.....	16
3.1 Infraestructura base seleccionada.....	16
3.2 Configuración del servicio proxy-web.....	17
3.2.1 Compilación y Configuración de Squid.....	17
3.2.2 Autenticación.....	19
3.2.3 Listas de control de acceso.....	25
3.2.4 Lista negra Malware.....	29
3.2.5 Inspección de tráfico cifrado.....	32
3.2.6 Antivirus de Navegación.....	37
3.2.7 Personalización de mensajes de error.....	39
3.3 Configuración del servicio de balanceo de carga.....	41
3.3.1 Instalación de un segundo nodo proxy.....	42
3.3.2 Instalación del servicio de balanceo de carga.....	43
4. Centralización de Logs.....	50
4.1 Descripción del servicios.....	50
4.2 Traspaso de los ficheros de logs.....	52
4.2.1 Instalación de Elasticsearch.....	52
4.2.2 Instalación de Logstash.....	53
4.2.3 Instalación de Beats.....	55
4.2.4 Instalación de Kibana.....	56
4.2.5 Formato de logs de Squid.....	57
4.3 Visualizaciones y Dashboard.....	58
5. Configuración de acceso de los usuarios.....	64
6. Implantación y servicios en Cloud.....	66
6.1. Funcionamiento de los productos elegidos en modo Cloud.....	66
6.2. Servicio DNS inteligente.....	66
CISCO UMBRELLA.....	67
TitanHQ.....	68

7. Conclusiones.....	69
8. Glosario.....	70
9. Bibliografía.....	71
10. Anexos.....	73
ANEXO 1: Script arranque servicio Squid.....	73
ANEXO 2: Respuesta unión con Kerberos, comando msktutil.....	75
ANEXO 3: Script actumalware.....	76
ANEXO 4: Código fuente de una página de error personalizada.....	77
ANEXO 5: keepalived.conf.....	77
ANEXO 6: logstash.conf.....	78
ANEXO 7: Visualizaciones y Dashboard de Kibana.....	78
ANEXO 8: Manual Operacional de la solución.....	81

## Lista de figuras

Figure 1: Planificación del Trabajo.....	4
Figura 2: Diagrama General.....	5
Figura 3: Apache Traffic Server.....	8
Figura 4: Haproxy.....	8
Figura 5: Squid-Cache.....	9
Figura 6: Balanceo de Carga.....	10
Figura 7: Interceptación HTTPS - Symantec Corporation.....	12
Figura 8: ClamAV Antivirus.....	13
Figura 9: SquidClamav.....	14
Figura 10: Elasticsearch.....	15
Figura 11: Logstash.....	15
Figura 12: Kibana.....	15
Figura 13: Diagrama General con Servicios.....	16
Figura 14: Autenticación del cliente con Active Directory.....	20
Figura 15: Peticiones del cliente al servidor proxy.....	20
Figura 16: Solicitud de Tiquet Kerberos.....	20
Figura 17: Respuesta de negociación.....	20
Figura 18: Petición cliente a proxy con autenticación LDAP.....	23
Figura 19: El cliente facilita sus credenciales.....	24
Figura 20: El servidor proxy verifica las credenciales contra el Active Directory.....	24
.....	24
Figura 21: El servidor proxy devuelve el contenido solicitado.....	24
Figura 22: Lista Negra Reyes CCN-Cert.....	30
Figure 23: Interacción Cliente Squid Web.....	32
Figure 24: Interacción Cliente Squid Web Cifrada.....	33
Figura 25: Intercepción SSL.....	33
Figura 26: Instalación Certificado Raiz Internet Explorer 11.....	35
Figura 27: Asistente de Importación de Certificados.....	36
Figura 28: Advertencia de Seguridad Importación de Certificados.....	36
Figura 29: Funcionamiento ICAP.....	37
Figura 30: Página de error Acceso Denegado.....	40
Figura 31: Página de Virus Detectado.....	41
Figura 32: Escalado Vertical.....	41
Figura 33: Escalado Horizontal.....	42
Figura 34: Cambio de roles - KeepAlived.....	44
Figura 35: Creación de SPN desde línea de comandos Windows.....	48
Figura 36: Contenido del fichero PROXY.keytab.....	49
Figura 37: Interacción ELK Stack.....	51
Figura 38: ELK Stack en los elementos.....	52
Figura 39: Página Web de respuesta Elasticsearch.....	53
Figura 40: Página principal de Kibana.....	57
Figura 41: Opción Visualize en Kibana.....	59
Figura 42: Visualización HTTP_METHOD.....	59
Figura 43: Visualización HTTP_STATUS_CODE.....	60
Figura 44: Visualización SQUID_REQUEST_STATUS.....	60
Figura 45: Visualización Peticiones por Proxy.....	61

Figura 46: Visualización Top 10 Web Solicitadas.....	61
Figura 47: Visualización Top 10 Usuarios.....	62
Figura 48: Opción Dashboard en Kibana.....	62
Figura 49: Dashboard en Kibana.....	63
Figura 50: Configuración en Internet Explorer 11.....	64
Figura 51: Configuración en Mozilla Firefox.....	64



# 1. Introducción

## 1.1 Contexto y justificación del Trabajo

En la actualidad, toda infraestructura TIC de tamaño medio o grande necesita de un sistema que controle y optimice la navegación por internet.

Los motivos por los que se debe implementar un sistema de este son muy variados. Desde el enfoque del rendimiento, dicho sistema optimiza el **consumo de ancho de banda** con el cacheo de las páginas web. Cuando un usuario solicita una página web, esta es almacenada en espera de que otro usuario la pueda solicitar. Si ello se produce, el servicio verificará si la página ha sido actualizada, si no es así la servirá desde su memoria cache. En caso contrario, la actualizará y servirá al cliente.

Por otro lado, desde el punto de vista de la seguridad, es bien sabido que el acceso a Internet es un **vector de ataque** utilizado por distintos software de tipo malicioso. Los usuarios acceden a páginas web de poca confianza y pueden infectar sus equipos. Los sistemas de filtrado de navegación web pueden solventar estos problemas gracias a dos estrategias.

La primera de ellas es el bloqueo de páginas webs con **listas negras**. Si el usuario no puede acceder a páginas no seguras verificadas por entidades de seguridad disminuirán las posibilidades de infección.

La segunda, es la implementación de un **antivirus de navegación**. Cuando el usuario navega, el sistema utiliza motores de análisis de antivirus para prevenir la infección del equipo.

Otro beneficio del sistema es la mejora del **rendimiento del trabajo**. En este caso, el sistema puede bloquear páginas web a las cuales la organización no quiera que sus empleados accedan dentro de su horario laboral. Por otro lado, pueden establecerse **distintos niveles de acceso**, dependiendo de los perfiles laborales que se encuentren en la organización. De tal forma que podemos tener usuarios con acceso libre, otros con acceso a ciertas paginas web y usuarios que tengan bloqueado el acceso.

Por otro lado, estos sistemas pueden prevenir las **fugas de información** y en caso de no poder evitarlo son capaces de ofrecer recursos forenses con sus logs. Gracias a ellos se puede analizar el comportamiento de los usuarios sospechosos y reproducir los accesos que realizaron.

En la actualidad se ha extendido el uso del protocolo seguro web HTTPS en la mayoría de las páginas webs. Al utilizar un protocolo de

comunicaciones seguro, los sistemas de de filtrado web han visto reducidas sus capacidades al no poder consultar el tráfico completo. Sólo son visibles las conexiones iniciales con las páginas, en las que se establece la conexión cifrada.

Ante esta situación, los sistemas de navegación han implementado métodos de **interceptación** en los que se descifra la comunicación para ser analizada. Este tipo de métodos son muy parecidos a lo que en seguridad informática se denomina ataque “man in the middle”. El sistema se encarga de mantener y controlar la comunicación cifrada entre ambos extremos, dándole acceso a todo el tráfico.

La Agencia de Gestión Agraria y Pesquera de Andalucía (en adelante AGAPA) ha considerado necesaria la implantación de un sistema de estas características y ha facilitado los medios para la realización de este proyecto.

## 1.2 Objetivos del Trabajo

Para la implementación de una solución de este tipo, se establece el siguiente listado de requisitos. Todos ellos deberán ser cubiertos para dotar al sistema de las funcionalidades y capacidades de un sistema como el descrito en el apartado anterior:

- Debe ser **escalable** en el número de nodos. El sistema debe permitir aumentar su capacidad sin merma en la calidad del servicio.
- Presentará distintos sistemas de **autenticación**. Debe proveer de al menos dos métodos de autenticación para identificar al usuario que realiza la navegación.
- Estará basado en **software libre**. Aunque existen soluciones de mercado, estas conllevan una inversión económica que se ve incrementada a la par que se amplían las distintas capacidades de dichos sistemas.
- Filtrado de navegación web. Debe ser capaz de **analizar el tráfico** web para dar acceso o no a los contenidos solicitados
- Establecerá distintos **niveles de acceso**. Debe permitir la creación de perfiles distintos de acceso a las páginas web según el usuario que realice la petición.

El proyecto no solo pretende cumplir con la lista de requerimientos indicada antes, sino que quiere dotar de herramientas que permitan controlar de manera efectiva el uso que se realiza del acceso a internet.

Siguiendo dicha premisa el proyecto establece los siguientes objetivos:

- Sistema de dos nodos proxy-web.
- Balanceador de carga entre los nodos.
- Sistema de autenticación por Kerberos y LDAP.
- Lista negra anti-malware actualizada.
- Antivirus de navegación.
- Mensajes de error personalizados.
- Inspección de tráfico SSL.
- Centralización de logs.

### 1.3 Enfoque y método seguido

Para la realización de este TFG se ha procedido a realizar una revisión de los principales productos open source que podían aportar una ventaja al sistema propuesto. Debido a que muchos desarrollos basados en software libre pueden llegar a abandonarse, se ha intentado utilizar productos con mayor nivel de madurez y popularidad. En general, los productos que cumplen estas características suelen poseer gran cantidad de documentación y recursos accesibles.

A la hora de implementar el sistema se ha optado por un enfoque que parte de la instalación de los elementos básicos para el sistema, para proseguir con cada una de las funcionalidades necesarias que se ha estimado conveniente incluir.

En concreto, el primer elemento que se configura del servicio es el servicio de proxy, básico para el sistema. Después, se incluyen las configuraciones necesarias para el funcionamiento deseado. Una vez alcanzado un nivel suficiente de madurez del sistema, se continúa con la configuración de alta disponibilidad. Para finalizar, se procede a configurar el sistema centralizado de Logs.

### 1.4 Planificación del Trabajo

La realización de este TFG se ha planificado siguiendo una serie de tareas o hitos a completar en un total de 15 semanas.

Las semanas iniciales se dedicaron a realizar la documentación necesaria relativa a la elección del tema de este TFG. En estas semanas también se acometió la planificación del mismo.

La planificación puede consultarse en la imagen siguiente.

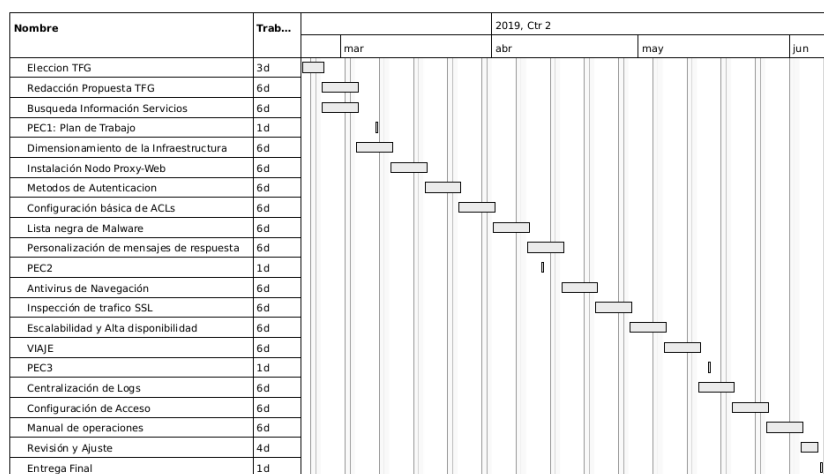


Figure 1: Planificación del Trabajo

## 1.5 Breve resumen de productos obtenidos

De la realización de este TFG se han obtenido el siguiente listado de productos y capacidades:

- Proxy Squid con las siguientes capacidades:
  - Alta disponibilidad.
  - Integración con Active Directory con Kerberos.
  - Inspección de tráfico SSL/TLS.
  - Lista negra de páginas malware actualizada.
- Balanceadores en alta disponibilidad Haproxy.
- Sistema centralizado de logs con Elasticsearch, Kibana y Logstash.

## 1.6 Breve descripción de los otros capítulos de la memoria

En los capítulos siguientes se comienza por describir la **solución diseñada a nivel general** sin especificar los componentes concretos. Después, se analizan para cada uno de los elementos necesarios, las alternativas disponibles para su implementación.

A continuación, se describe la implementación de la solución para cada uno de los elementos. Se dedica un apartado independiente para el sistema centralizado de logs.

Para finalizar, se incluye un apartado dedicado a las configuraciones necesarias en la parte cliente del sistema. Se propone una estrategia de configuración de los mismos en el entorno donde se ha implementado el sistema. Se incorpora un pequeño manual de operación del sistema que incluye las tareas más usuales en el mantenimiento de esta solución.

## 2. Diseño y elección de componentes del sistema.

### 2.1 Diseño del sistema

En la figura siguiente se muestra el diagrama general del sistema con los distintos elementos que lo componen. En los siguientes apartados se describirán los mismos, indicando las características y capacidades que los hacen ser idóneos para ser utilizados en este sistema. Se indicarán soluciones alternativas o complementarias, así como sus capacidades.

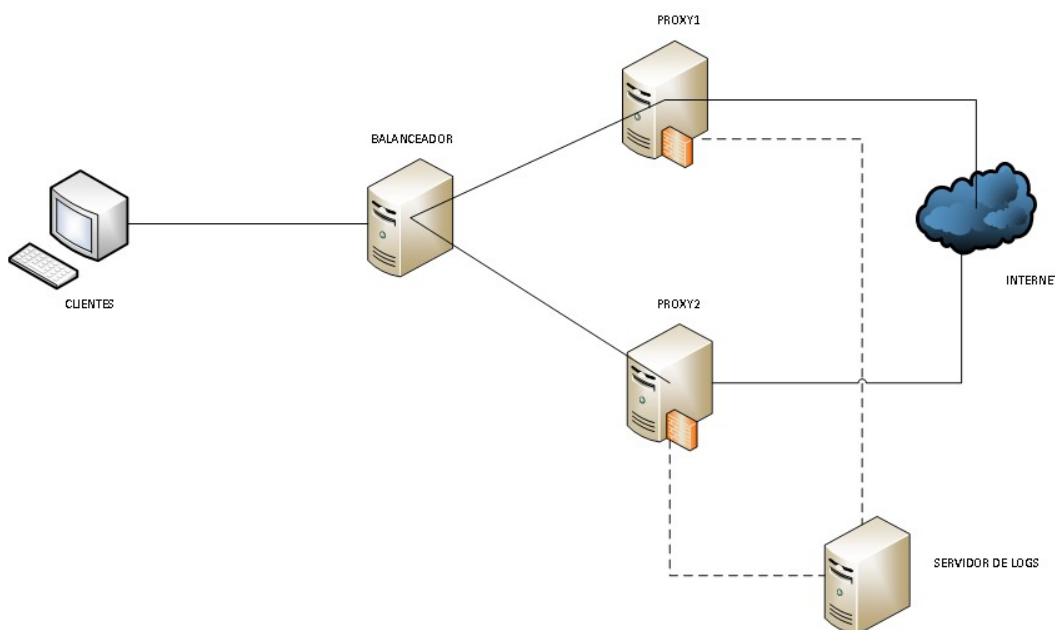


Figura 2: Diagrama General

Como se puede ver en la figura anterior, este sistema consta de 3 elementos diferenciados. El primero, más importante y pieza fundamental de este TFG, es el servicio de **proxy-web**. Este será el encargado de intermediar entre los clientes y las páginas web. También ejercerá el filtrado web, análisis del tráfico cifrado y motor de antivirus. Estos dos últimos serán analizados en apartados separados debido a su especial importancia.

Otro de los elementos es el sistema de **balanceo de carga**. Ya que el sistema debe ser escalable, debe existir en la infraestructura un elemento que reparta las peticiones entre los distintos nodos que se pueden ir añadiendo al sistema. En el apartado dedicado a este tema se analizarán las principales soluciones que existen en el mercado para realizar esta función.

El tercer elemento que aparece en el diagrama es el sistema de **centralización de logs**. Como se ha indicado antes, un recurso esencial para la seguridad de la información es conocer quién y cuando

se ha accedido a la misma. Para ello se necesita de un sistema que almacene los logs de los distintos nodos proxy-web para que puedan ser consultados de forma ágil y sencilla. Al igual que los elementos descritos antes, se analizarán los sistemas y soluciones que se encuentran disponibles.

## 2.2 Servicio de proxy-web

### ¿Qué es un servicio proxy?

El primer elemento del sistema y pieza fundamental del mismo es el servicio de proxy-web. Este sistema es el **intermediario** entre los usuarios de una red y el acceso a internet. De manera general, estos sistemas funcionan según el proceso siguiente:

- El cliente solicita una página web al servidor proxy.
- El servidor proxy recibe la petición y consulta si la información solicitada la tiene guardada en su memoria cache.
- Si dispone de ella verifica si está actualizada, si es así se la ofrece al cliente.
- Si no es así, actualiza la información y se la ofrece al cliente.
- En caso de no disponerla en la memoria cache, la solicita, la almacena y se la ofrece al cliente.

Los servidores proxy-web también suelen denominarse proxy-cache ya que una de sus principales funciones es el almacenamiento de contenido par ahorrar ancho de banda y aumentar la velocidad de respuesta.

### Modos de funcionamiento

Estos servicios tienen distintos modos de funcionamiento, como son:

- Transparente. Este modo no se configura en los navegadores o programas cliente. Por defecto, el tráfico web es remitido al servidor proxy que será el encargado de suministrar el contenido.
- Directo o Explicito. Este modo requiere que en los clientes se configure la dirección del servicio para poder funcionar.
- Inverso. Este modo, como su nombre indica, realiza el proceso contrario al explicado antes. Cuando un cliente solicita una web, el proxy es capaz de direccionarlo a distintos *backend* según su programación.

## Tipos de Proxy

También se pueden diferenciar los servicios proxy según el protocolo para el que realizan la intermediación:

- Proxy HTTP. Utilizado para el cacheo de páginas web y ficheros ofrecidos con protocolo HTTP.
- Proxy SSL. Este tipo de proxy es el encargado de manejar peticiones SSL o *Secure Sockets Layer*, entre el cliente y los servidores web. El protocolo SSL es utilizado para asegurar el tráfico de datos. También se les suele denominar Proxy HTTPS. Este punto es tratado en mayor profundidad en el apartado de descifrado SSL.
- Proxy Socks. El protocolo SOCKS (diminutivo de SOCKeT\$) es un protocolo utilizado para la comunicación entre clientes y servidores de manera transparente a los dispositivos de seguridad de la red.
- Proxy FTP. Utilizado para cachear ficheros descargados con protocolo FTP.

## Soluciones proxy-cache privativas.

Dado que una de los principales requisitos de este TFG es la necesidad de que las aplicaciones utilizadas en el sistema sean software libre, sólo se analizarán de forma breve las soluciones comerciales disponibles en el mercado.

- Integrados en dispositivos firewall.  
Muchos de los productos Firewall que se encuentran en el mercado incorporan la funcionalidad de filtrado web. En concreto este tipo de dispositivos suelen funcionar como **proxy en modo transparente**. Fabricantes como Fortinet, Sophos, Palo Alto o Checkpoint lo incorporan en sus dispositivos firewall e incluso pueden funcionar como modo directo o explícito. En estos casos es una funcionalidad adicional a la propia del dispositivo por lo que el precio de los mismos puede ser muy elevado si sólo es necesaria la función de proxy.
- Symantec ProxySG and Advanced Secure Gateway. Este software fue desarrollado por la compañía BlueCoat, En la actualidad, forma parte del portafolio de productos de la empresa Symantec.  
Este producto se vende como un *appliance* instalable en la infraestructura IT de la organización para ofrecer el servicio. Cubre todos los requerimientos necesarios para la implantación en este proyecto, pero tiene un coste que variará dependiendo de

la carga que deba soportar el dispositivo. Para más información puede consultarse la web del fabricante [1].

### Soluciones proxy open source

- Apache Traffic Server



*Figura 3: Apache Traffic Server*

Este software es un proxy-cache que en sus versiones primeras fue de tipo comercial, después fue adquirido por Yahoo y para finalizar, donado a la Apache Software Foundation bajo Apache License, Versión 2.0.

Puede funcionar como proxy-cache en modo directo, transparente e inverso. Se ha construido de modo que puedan añadirse funcionalidades con *plugins*. Se encuentra en desarrollo la funcionalidad de balanceador de carga.

La página web oficial del producto [2] ofrece una guía de referencia para la traducción de las configuraciones de Squid a Apache Traffic Server.

Una de los contra de este software es que a día de hoy **no cuenta con un sistema de soporte de fabricante**.

Otra funcionalidad no detectada en la herramienta es la capacidad de inspeccionar tráfico cifrado.

Por otro lado, no se encuentra documentación sobre distintas implementaciones de autenticación para este sistema. Todo hace indicar que se podría habilitar con la inclusión de un plugin que facilitase dichas capacidades. A día de hoy no se encuentra dicho componente en un estado estable.

- Haproxy



*Figura 4: Haproxy*



Este software proxy tiene como principal función el **modo inverso y el balanceo de carga** entre nodos para aplicaciones HTTP y TCP.

Por contra a lo anterior, no tiene funciones de cache y su funcionalidad como proxy web en modo directo es casi inexistente.

Para mas información puede consultarse la web del fabricante [3].

- Privoxy

Este servicio proxy es un proyecto open source desarrollado por una comunidad de usuarios y provee versiones para Linux y Windows. Según indica la página oficial del proyecto [4], **carece de funcionalidad de cache** y tampoco es capaz de interceptar comunicaciones SSL.

- Nginx [5] o Varnish-Cache [6]

Estas dos soluciones de software son muy potentes pero están destinadas a funcionar como **proxy inversos** con funciones de **cache**, carecen de funcionalidad en modo directo o *forward*, esta última es la indicada para nuestra solución.

- Squid-Cache



Figura 5: Squid-Cache

Este software, desarrollado por la *Squid Software Foundation*, fue lanzado en 1996 y aún sigue en evolución. Es uno de los software mas populares dentro de su tipo y se encuentra licenciado con GNU GPL.

Al llevar tanto tiempo en servicio, existe multitud de recursos de información a los que acceder y su funcionalidad ha ido creciendo de manera constante. Así mismo, existe una **amplia bibliografía** sobre su funcionamiento, como por ejemplo el libro *Squid: The Definitive Guide* [7] o la guía *Squid Proxy Server 3.1: Beginner's Guide* [8]. Además, cuenta con una wiki de acceso público con ejemplos de configuración [9].

Este servicio soporta los protocolos HTTP, HTTPS, FTP y GOPHER. Está construido basado en un programa principal (squid) mas programas adicionales encargados, por ejemplo, de

reescritura de peticiones, gestión de autenticaciones o herramientas de administración.  
Dispone de versiones para múltiples sistemas operativos tanto windows, linux, BSD, unix y OS/2.

Permite establecer **jerarquías de cache** para poder construir estructuras de almacenamiento complejas para acelerar el acceso a los datos. Para ello sigue protocolos como HTCP, CARP, ICP, y caché digests utilizados para la consulta de objetos a cache.

Squid puede funcionar como proxy directo, transparente y también como proxy inverso con capacidades de acelerador web.

Debido a que Squid cubre con sus capacidades todos los requisitos establecidos para este componente del sistema, unido a la estabilidad y experiencia del mismo, es el software escogido este TFG.

### 2.3 Servicio de balanceo de carga

Para poder garantizar la escalabilidad de la solución, es necesario incorporar un servicio de balanceo que pueda repartir la carga entre distintos nodos proxy.

En este caso, el servicio de balanceo de carga **no presenta unos requisitos elevados**. En principio, sólo es necesario balancear el servicio por IP y puerto por el que se ofrecerá el servicio de proxy.

En concreto, el balanceador ofrecerá una IP de frontend por el puerto 3128 (puerto por defecto del servicio proxy) y enviará las peticiones a los backend proxy por el mismo puerto. Debido a esto, la mayoría de los servicios de balanceo de carga open source que existen en la actualidad pueden cubrir dicha necesidad.

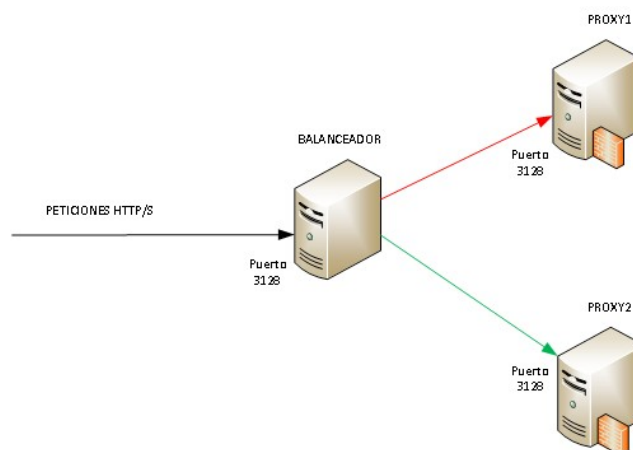


Figura 6: Balanceo de Carga

En el caso concreto de este TFG, la organización donde se va a implantar la solución, ya hace uso de servicios de este tipo con el aplicativo **Haproxy** descrito en el apartado anterior. De tal forma que por motivos de homogeneidad del servicio se implantará como balanceador de carga para la solución.

En cualquier caso, se incluye un listado de productos open source y privativos que pueden ofrecer el servicio que se necesita:

- Nginx [5]
- Seesaw [10]
- LoadMaster by Kemp [11]
- Zevenet [12]
- Neutrino [13]
- Balance by InLab [14]
- Traefik [15]

Por otro lado, el mercado ofrece soluciones de balanceo de carga con **hardware** appliance:

- F5 [16]
- Barracuda [17]
- TP-Link [18]
- Fortinet [19]

## 2.4 Servicio de descifrado SSL

Un servicio proxy recibe las peticiones HTTP que realizan los clientes. Este servicio seguirá toda la traza de peticiones y comunicaciones entre el cliente y el servidor web de destino y los registrará en su log, y aplica las políticas de acceso que se hayan establecido.

Al ser un protocolo no cifrado, el análisis de las URLs puede llegar a niveles mas profundos que el dominio de la petición web, de tal forma que **puede reconstruirse el comportamiento del navegador** en la página web que se solicitó.

El protocolo HTTPS es una implementación segura de HTTP con cifrado de la comunicación entre las partes. En este caso, el cliente establece como primer paso, un canal seguro de comunicación con el

servidor web de destino. Los servidores proxy sólo registran esta conexión inicial como un **CONNECT** del protocolo al dominio web al que se conecta. El resto de la comunicación que pasa por el proxy no queda registrada y escapa del análisis. Dada esta situación **sólo se puede conceder o denegar el acceso** al dominio al que se quiere conectar.

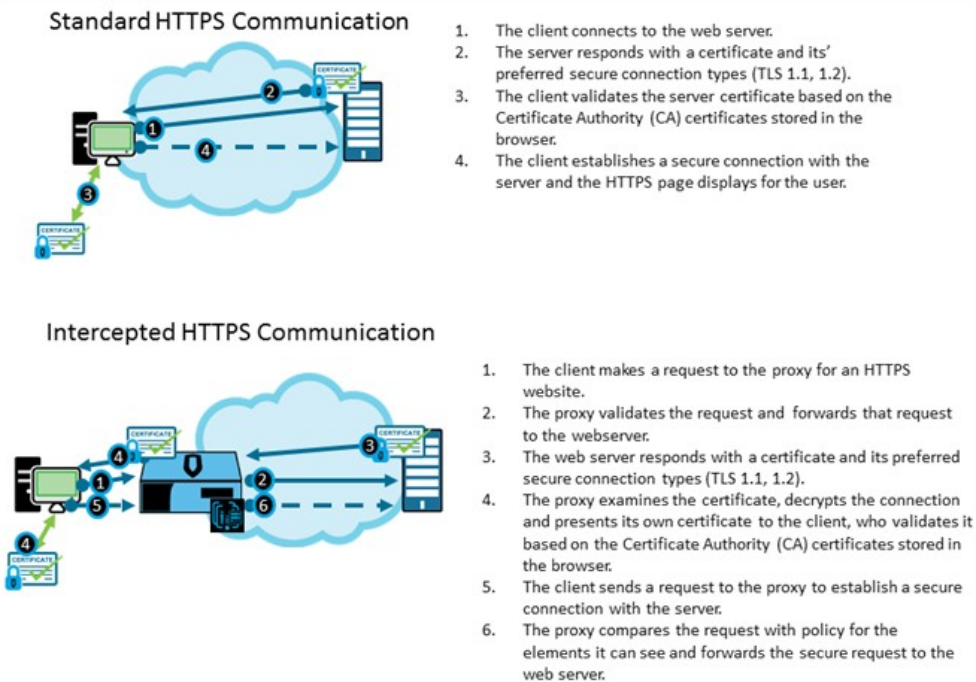


Figura 7: Intercepción HTTPS - Symantec Corporation

En la actualidad, la aplicación del protocolo HTTPS se está expandiendo y los grandes portales web como Google, ya han implantado HTTPS como el protocolo por defecto para acceder a sus servicios.

Ante esta situación, tanto los servidores proxy como los firewall de nueva generación han implementado la **funcionalidad de interceptación SSL/TLS**. Con ella, los dispositivos descifran la comunicación entre cliente y servidor para poder analizar el tráfico. Esta funcionalidad ha supuesto un aumento en el gasto de recursos de los servicios así como un retardo en el tiempo de respuesta.

La estrategia que se aplica es similar a las técnicas de ataque **Man-in-the-Middle (MiTM)** solo que en este caso debe ser consentida y aprobada. El servicio proxy establecerá conexiones seguras con ambos miembros de la comunicación.

Esta funcionalidad tiene una implicación ética y legal ya que se vulnera el propósito de la utilización de un protocolo de comunicación segura como es HTTPS. En el caso que ocupa este TFG, las ventajas de realizar esta función superan los inconvenientes.

En primer lugar, en los últimos años se ha detectado un aumento del uso del protocolo HTTPS por software malware y virus como **vectores de ataque e infección**.

En segundo lugar, el **uso inadecuado** de recursos web queda oculto con el uso de dicho protocolo.

Por último lugar, si no se implementa esta función los logs del sistema ofrecen **poca información** en caso de requerir un análisis forense del tráfico realizado.

En el apartado donde se describe la implementación de esta funcionalidad se amplía el funcionamiento concreto dentro del sistema proxy, así como los requisitos necesarios para su implementación en la parte cliente.

Ante todo lo anterior, es necesario realizar la comunicación del uso de esta funcionalidad a los usuarios de la red que van a hacer uso de este sistema.

## 2.5 Servicio de Antivirus de Navegación

Una vez ha quedado determinado ofrecer el servicio de proxy con el software Squid, queda analizar cómo se puede implementar un escaneado de virus sobre el tráfico de navegación.

Un primer estudio de las soluciones disponibles nos muestra que el motor de antivirus más utilizado junto con squid es ClamAV [20]. Este producto es Open source bajo licenciamiento GPL.

Por otro lado, mantiene una base de datos actualizada con definiciones de virus y soporta diferentes formatos como ficheros MacOffice, Microsoft Office, Flash, HTML, RTF y PDF. También puede escanear diferentes formatos de ficheros comprimidos.



*Figura 8: ClamAV Antivirus*

SquidClamav [21] es el **redirector de tráfico** para squid y se integra como servicio ICAP (Protocolo de Adaptación de Contenidos de Internet)



*Figura 9: SquidClamav*

Una limitación que se indica de esta solución afecta a la hora de escanear ficheros descargados de gran tamaño en caliente. En este caso, el servicio no puede escanearlo mientras se descarga, por lo tanto, realiza la descarga en el servidor proxy y después es escaneado. Si el antivirus detecta una amenaza redirigirá la petición del cliente a una página web personalizable. En caso de no encontrarse amenazas se enviará el fichero al cliente.

Otra solución disponible es squid-vscan de OpenAntivirus.org. Este software también se integra con Squid para dotar de análisis del tráfico de navegación web. En el momento de la realización del proyecto este producto se encuentra con el **desarrollo congelado** en la versión 1.0 desde el año 2002.

Viralator es otra alternativa antivirus integrable en Squid y licenciada bajo GPL. Este producto se ha desarrollado como script en lenguaje Perl. Esta solución **no incluye el motor de antivirus** por lo que es necesario incluir uno en la solución. Según su web oficial [22], acepta múltiples motores tanto comerciales como open source. Su funcionamiento se basa en realizar la descarga previa de los ficheros, analizarlos y si no contienen virus, lo sirve al cliente que lo solicitó. En el momento de la realización del proyecto este producto se encuentra con el **desarrollo congelado** desde el año 2013.

Entre las opciones analizadas, sólo Clamav ha ofrecido garantías suficientes para ser incluido en el sistema. Las otras dos alternativas se han detectado como desarrollos abandonados y en versiones iniciales.

## 2.6 Centralización de logs

El último componente del sistema a incluir es el encargado de recolectar los logs de los nodos proxy para su posterior análisis.

En este caso, la solución open source más utilizada es la denominada Elastic Stack [23] (antes ELK), la cual es la combinación de tres productos, Elasticsearch, Logstash y Kibana.



*Figura 10: Elasticsearch*

Elasticsearch [24] es un motor de búsqueda y análisis en grandes cantidades de datos. Está basado en Lucene [25], desarrollado en Java y bajo licencia Apache.



*Figura 11: Logstash*

Logstash [26] es el componente encargado del traslado de los ficheros de logs desde los servidores backend para que sean accesible por Elasticsearch.



*Figura 12: Kibana*

El tercer componente es Kibana [27]. Es el encargado de facilitar la visualización de los datos gestionados por los otros dos componentes. Con Kibana se puede implementar un *Dashboard* que facilite la interpretación de los logs.

## 3. Implementación de la solución

### 3.1 Infraestructura base seleccionada

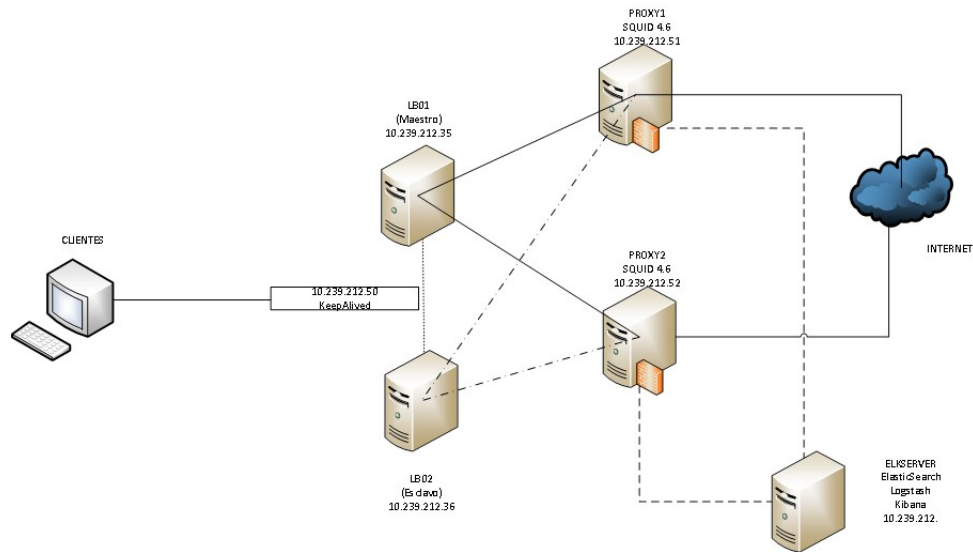


Figura 13: Diagrama General con Servicios

El sistema se construye con dos balanceadores de carga, LB01 y LB02. El primero de ellos actúa como Maestro y el segundo como Esclavo, queda a la espera por si el primero dejase de prestar el servicio. Para ello se implementa el servicio keepalived por el cual se **comparte una dirección IP** entre ambos, la cual será por la que se publicará el servicio a balancear. Dicha configuración se especifica en el apartado 3 de este capítulo. Ambos balanceadores tienen instalado el servicio Haproxy en su versión 1.8.

El siguiente nivel de respuesta son los proxy-cache PROXY1 y PROXY2. En este caso, ambos nodos responden de forma activa a las peticiones derivadas por los balanceadores de carga. Estos elementos son los encargados de realizar las funciones que se describen en los apartados que forman parte del capítulo 3.2. Los servidores PROXY1 y PROXY2 utilizan como servicio base Squid 4.6.

El tercer nivel es el sistema de recolección de logs del sistema. Para este sistema se dispone de un servidor denominado ELKSERVER, el cual integra los tres servicios que forman parte del sistema, Elasticsearch, Logstash y Kibana. La configuración de estos elementos se describe en el capítulo 4 de este documento.

La elección de estos componentes viene determinada por el hecho de que este mismo sistema es utilizado ahora por la AGAPA como sistema de recolección de logs de aplicaciones web. Debido a esto, se opta por **homogeneizar** la instalación y aprovechar los conocimientos y experiencia ya adquiridos en la herramienta.



Todos los servidores que forman parte de este sistema se han instalado con sistema operativo **Debian 9.8**. La elección de dicho sistema es debido a la política que sigue AGAPA a la hora de utilizar distribuciones de sistemas operativos Linux.

## 3.2 Configuración del servicio proxy-web

En este capítulo se describen las distintas configuraciones realizadas en el servicio proxy para satisfacer los requisitos planteados.

### 3.2.1 Compilación y Configuración de Squid

En este apartado se indica la configuración elegida para la instalación del servicio Squid así como los parámetros más importantes de la configuración general del servicio y los motivos que llevan disponer de la configuración elegida de los mismos.

Para la instalación del servicio de Squid se hace necesario la descarga del código fuente del mismo y su **compilación**. Uno de los motivos es habilitar ciertas características que por defecto no vienen activas en la instalación por defecto.

El primer paso es instalar los paquetes del sistema operativo que son necesarios para el funcionamiento correcto de Squid. Para ello se ejecuta en el interprete de comandos:

```
> apt-get install gcc binutils make
> apt-get -y install openssl devscripts build-essential fakeroot libdbi-perl libssl1.0-dev
```

Una vez se ha realizado la instalación de los mismos podemos proceder a la compilación de squid. Para ello nos descargamos los fuentes del software de la web del proyecto [28].

Descomprimos el archivo descargado y ya podemos proceder a la compilación del programa. Para ello primero debemos indicar los parámetros con los que se compilará. En caso de querer saber los **parámetros** que acepta Squid se puede lanzar el siguiente comando:

```
> ./configure --help
```

A continuación se muestra la línea de compilación lanzada.

```
> ./configure --prefix=/opt/squid46 --includedir=/include --mandir=/share/man --infodir=/share/info --localstatedir=/opt/squid46/var --disable-maintainer-mode --disable-dependency-tracking --disable-silent-rules --enable-inline --enable-async-io --enable-storeio=ufs,aufs,diskd --enable-removal-policies=lru,heap --enable-delay-pools --enable-cache-digests --enable-underscores --enable-icap-client --enable-follow-x-forwarded-for --enable-auth --enable-digest-auth-helpers --enable-negotiate-auth-helpers --enable-auth-ntlm --enable-arp-acl --enable-esi --disable-translation --with-logdir=/var/log/squid46 --with-pidfile=/var/run/squid46.pid --with-filedescriptors=65536 --with-large-files --with-default-user=proxy --enable-linux-netfilter --enable-ltdl-convenience --with-openssl --enable-ssl --enable-ssl-crt
```

De todos los parámetros utilizados se destacan los siguientes:

- `--prefix=/opt/squid46` → Ubicación de la instalación.
- `--enable-icap-client` → Habilitar el cliente icap, necesario para antivirus.
- `--enable-follow-x-forwarded-for` → Habilitar x-forward.
- `--enable-auth --enable-digest-auth-helpers --enable-negotiate-auth-helpers --enable-auth-ntlm` → Habilita distintos sistemas de autenticación.
- `--with-default-user=proxy` → Usuario con el que se ejecutará squid.
- `--with-openssl --enable-ssl --enable-ssl-crt` → Necesario para la inspección SSL.
- `--with-logdir=/var/log/squid46` → Carpeta donde se generarán los logs de Squid.

Una vez finalizado el comando “configure” se procede con la compilación de squid.

```
> make && make install
```

Una vez finalizada la compilación resta habilitar los **permisos** de escritura de los logs en la carpeta indicada en la compilación (en caso de que no exista deberá crearse).

```
> chown proxy.proxy /var/log/squid46
```

En este punto puede ejecutarse el servicio de squid con el siguiente comando:

```
> /opt/squid46/sbin/squid
```

Para que el servicio funcione de manera correcta deberá configurarse el fichero `squid.conf`, en el que se especifican los parámetros principales del servicio. El fichero se encuentra en la carpeta `/opt/squid46/etc`.

A continuación se presentan alguno de los parámetros de configuración del fichero `squid.conf` que se han introducido.

- `visible_hostname proxy1.dap.es`  
Nombre por el que se identifica el servidor.
- `http_port 3128`  
Puerto por el que escucha el servicio proxy.
- `debug_options ALL,1 33,2 28,9`  
Para habilitar el modo avanzado de logs.
- `cache_dir ufs /var/spool/squid46 15360 16 256`  
La carpeta donde se creará la estructura de cache de squid.
- `maximum_object_size 8192 KB`

Tamaño máximo de objetos que se almacenará en cache.

- **Logformat**  
Definición del formato de logs.
- **max\_filedesc 4096**  
Número máximo de ficheros abiertos de la cache.
- **query\_icmp on**  
**icp\_port 3130**  
Puerto de escucha para consultas a la cache del servidor.
- **error\_directory /opt/squid46/etc/errores**  
Directorio donde se almacenan las páginas html que devolverá squid en caso de error.
- **forwarded\_for on**  
Para que el proxy no sustituya la IP del cliente cuando se haga una petición por el mismo.

Dado que la instalación del servicio de squid se ha realizado con la compilación de las fuentes, no se dispone de un **script de arranque** y parada del servicio. En nuestro caso se hará uso de un script de arranque modificado para esta instalación que se puede consultar en el ANEXO 1 y que se copiará a la ruta /etc/init.d.

### 3.2.2 Autenticación

Tal y como se ha indicado en el apartado de requisitos del sistema a implantar, el mismo debe disponer de **dos sistemas** que identifiquen a los usuarios que realizan la navegación.

En el caso concreto de la AGAPA, se dispone de un sistema de directorio basado en **Microsoft Active Directory**, el cual mantiene la base de datos de usuarios y contraseñas de los equipos clientes de la red. El domino DNS por el que responde es "DAP.ES".

Por defecto los equipos cliente unidos al dominio de Active Directory verifican sus credenciales con **Kerberos** para acceder a los recursos de la red. Dado esto, al integrar el sistema proxy con la base de datos y contraseñas de Active Directory no es necesario duplicarla y los usuarios no necesitan recordar dos credenciales.

Por otro lado, si se utiliza el sistema Kerberos, el usuario al tener ya un ticket activo de autenticación, no necesita proveer de credenciales a squid para verificar su identidad. Sólo debe verificarse la validez del ticket.

A continuación se muestran en imágenes el proceso que seguiría una petición dentro del sistema de autenticación con Kerberos.

1. El cliente provee al sistema de su usuario y contraseña, el sistema devuelve el resultado de la autenticación.

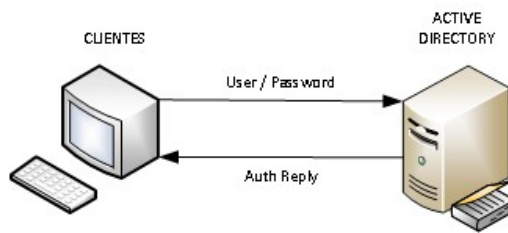


Figura 14: Autenticación del cliente con Active Directory

2. El cliente realiza peticiones de navegación al servidor proxy, el cual le responde con la fase de negociado de credenciales.

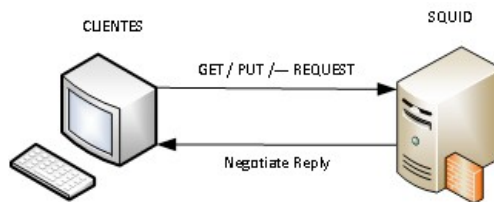


Figura 15: Peticiones del cliente al servidor proxy

3. El cliente solicita un tiquet Kerberos al servidor de Active Directory.

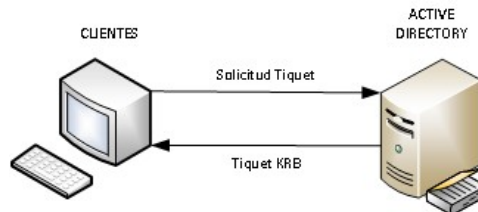


Figura 16: Solicitud de Tiquet Kerberos

4. El cliente responde a la negociación de Squid, el cual verifica la validez del usuario y aplica las ACL.

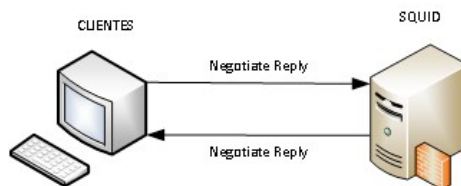


Figura 17: Respuesta de negociación

A continuación se especifica el proceso de configuración para que el servidor proxy sea capaz de utilizar Kerberos como método de autenticación.

- Para que el sistema pueda validar debe estar sincronizado con la misma hora que el servidor de Active Directory. Para ello se debe configurar el servicio NTP (Network Time Protocol).

Instalación del servicio cliente de conexión.

```
>apt-get install ntp ntpdate
```

Añadir a /etc/ntp.conf

```
> server IP_SERVIDOR_NTP
> restrict IP_SERVIDOR_NTP mask 255.255.255.255 nomodify notrap noquery
```

- Dar de alta en los DNS de Active Directory los registros A y PTR del servidor proxy.
- Instalar los paquetes necesarios para permitir la autenticación (se incluyen los necesarios para la autenticación con LDAP que se describe mas adelante).

```
> apt-get install krb5-user libkrb5-dev libsasl2-modules-gssapi-mit libsasl2-modules dnsutils
libsasl2-dev
```

```
> apt-get install libldap2-dev libsasl2-modules-ldap libsasl2-modules-gssapi-mit ldap-utils
```

```
> apt-get install samba build-essential winbind samba-common-bin
```

(Se recuerda que estos son los paquetes necesarios en una distribución Linux Debian 9 y correspondiente al tiempo de la redacción de este TFG).

- Es necesario configurar el sistema kerberos para que conecte con los servidores de Active Directory. Para ello se edita el fichero krb5.conf como el que sigue:

```
[libdefaults]
default_realm = DAP.ES
dns_lookup_kdc = yes
dns_lookup_realm = yes
ticket_lifetime = 24h

default_tgs_enctypes = aes256-cts-hmac-sha1-96 rc4-hmac des-cbc-crc des-cbc-md5
default_tkt_enctypes = aes256-cts-hmac-sha1-96 rc4-hmac des-cbc-crc des-cbc-md5
permitted_enctypes = aes256-cts-hmac-sha1-96 rc4-hmac des-cbc-crc des-cbc-md5

[realms]
DAP.ES = {
    kdc = dc1.dap.es
    kdc = dc2.dap.es
    admin_server = dc1.dap.es
}

[domain_realm]
.dap.es = DAP.ES
dap.es = DAP.ES
```

- Probamos que el sistema conecta contra los servidores Kerberos:
  - Creamos el tiquet Kerberos:

> kinit administrador@DAP.ES ← Introducimos la contraseña del usuario de Active Directory.

- Listamos los tiquets Kerberos:

> klist → Nos devuelve:

```
ticket cache: FILE:/tmp/krb5cc_0
  Default principal: administrador@DAP.ES
Valid starting Expires Service principal
11/03/19 12:59:12 11/03/19 22:59:12 krbtgt/DAP.ES@DAP.ES
renew until 12/03/19 12:59:09
```

- Instalamos el software mskutil que nos permitirá gestionar el fichero de tiquet Kerberos así como la unión del servidor proxy al dominio de Active Directory.

> apt-get install mskutil

- Creamos el fichero keytab que almacenará los tiquets Kerberos y unimos el servidor al dominio.

```
> mskutil -c -b "OU=SRV_LINUX" -s HTTP/proxy1.dap.es -k /opt/squid46/etc/PROXY.keytab --
computer-name PROXY1-K --upn HTTP/proxy1.dap.es --server dc1.dap.es --verbose --enctypes 28
```

Se identifican en la sentencia la ubicación en el arbol de directorio activo donde se creará la cuenta del servidor "OU=SRV\_LINUX", luego el nombre de servicio "HTTP/proxy1.dap.es", el nombre del fichero a generar "/opt/squid46/etc/PROXY.keytab", el nombre con el que se creará el objeto en Active Directory "PROXY1-K", el User Principal Name "HTTP/proxy1.dap.es", el servidor contra el que se conectará y el tipo de encriptado de la comunicación, en este caso necesario, al tratarse de un Active Directory en versión 2008 o superior.

El sistema debe devolver una respuesta similar a la que aparece en el Anexo 2.

- Una vez creado el fichero keytab, debemos dar acceso de lectura al servicio de squid.

```
> chgrp proxy /opt/squid46/etc/PROXY.keytab
```

```
> chmod g+r /opt/squid46/etc/PROXY.keytab
```

- Instalación de helper para Kerberos

El sistema de autenticación de squid se basa en un sistema base de negociación que hará uso de **programas específicos** del sistema con el que se quiere autenticar. En el caso de Kerberos utilizaremos el software Squid Kerberos Authentication Helper [29].

- Descargamos el software, descomprimos y compilamos:

```

> tar -xvf squid_kerb_auth-1.0.7.tar.gz
> cd squid_kerb_auth-1.0.7
> ./configure
> make && make install

```

- Se agrega al fichero de configuración de squid (squid.conf) la siguiente configuración para que pueda realizar la consulta kerberos.

```

auth_param negotiate program /opt/squid46/libexec/negotiate_wrapper_auth --ntlm
/usr/bin/ntlm_auth --diagnostics --helper-protocol=squid-2.5-ntlmssp --domain=DAP --kerberos
/usr/local/bin/squid_kerb_auth -i -r -s GSS_C_NO_NAME

```

```

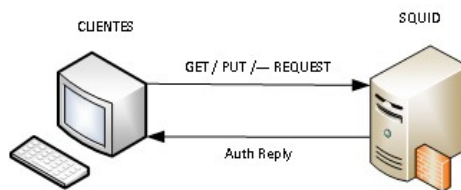
auth_param negotiate children 100
auth_param negotiate keep_alive on

```

En estas linea se establece los programas de autenticación que se utilizan, el protocolo Kerberos y un segundo protocolo de autenticación utilizado por Microsoft llamado **NTLM** y que utilizará en caso de no poder hacer uso de Kerberos. Se establecen el número de “hijos” que puede crear para facilitar la autenticación y que garanticen la rapidez necesaria.

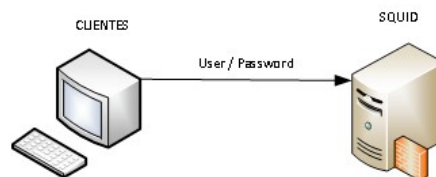
Una vez ya disponemos del primer sistema de autenticación para el servicio de proxy se configura el **segundo método** por el protocolo **LDAP**. Para ello, ya se han instalado antes los paquetes y dependencias necesarias. En este caso, cuando se le solicite al proxy una página web, el sistema lanzará una **ventana popup** solicitando las credenciales. El proceso queda descrito con las siguientes imágenes.

1. El cliente realiza una petición al servidor proxy. Este le solicita las credenciales de acceso.



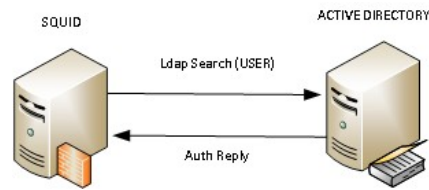
*Figura 18: Petición cliente a proxy con autenticación LDAP*

2. El cliente facilita las credenciales al servidor.



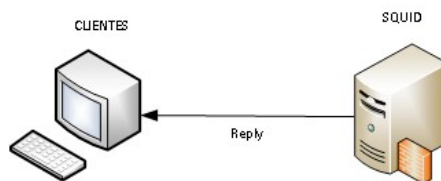
*Figura 19: El cliente facilita sus credenciales*

3. El servidor valida las credenciales contra el servidor LDAP (en nuestro caso Active Directory) y obtiene la respuesta de este.



*Figura 20: El servidor proxy verifica las credenciales contra el Active Directory*

4. Si el resultado ha sido exitoso el proxy devolverá el contenido solicitado.



*Figura 21: El servidor proxy devuelve el contenido solicitado*

A continuación se indica el proceso de configuración para que squid permita este método de autenticación.

- Añadimos la siguiente configuración en el fichero squid.conf

```
auth_param basic program /opt/squid46/libexec/basic_ldap_auth -P -R -b "dc=dap,dc=es" -D "cn=apache,cn=Users,dc=dap,dc=es" -W /opt/squid46/etc/ldappass.txt -f sAMAccountName=%s -h dc1.dap.es
```

```
auth_param basic children 30
auth_param basic realm Proxy AGAPA
auth_param basic credentialsttl 1 hour
```

En dicha configuración se especifican parámetros como, la ruta base del directorio LDAP desde el que se realizará la búsqueda, el usuario con el que se conectará squid (en este caso el usuario "apache") y un fichero que tiene almacenada la contraseña de este usuario, el campo de Active Directory por el que identificará al usuario (en caso de Active Directory es el campo "sAMAccountName", en caso de LDAP, en general es "uid"), un mensaje identificativo y el tiempo de validez de la credencial suministrada. El último campo es el servidor al que se realizará la consulta.

- Es necesario crear el fichero ldappass.txt con la contraseña del usuario "apache" que es el que realiza la consulta.
- Sobre este fichero hay que dar acceso al usuario "proxy" para que pueda consultarlo.



Para que el sistema requiera a los usuarios de sus credenciales se deberá incluir dicha condición en forma de acl en el fichero squid.conf como la siguiente:

```
> acl Usuarios_Autenticados proxy_auth REQUIRED
> http_access allow Usuarios_Autenticados
```

Una vez implementados ambos sistemas de autenticación, el servidor Squid aplicará un **procesamiento secuencial**. Este depende del orden en que se establezcan los sistemas dentro del fichero de configuración.

En este caso, se establece como sistema primario o principal la autenticación mediante Kerberos al colocarlo antes de la autenticación mediante Ldap. Así cuando un cliente solicite una página web se activará la autenticación por Kerberos. En caso de que no consiga una autenticación válida mediante dicho sistema, pasará a intentar realizar la autenticación con Ldap.

Un ejemplo de este funcionamiento aparece cuando un equipo no pertenece al dominio de Active Directory pero tiene configurado el proxy. Dicho equipo, no posee un tiquet válido en el Active Directory, por lo tanto, el sistema no lo autenticará mediante este método y proseguirá con el siguiente. En este caso, una vez descartado en método anterior, se activará el siguiente en el fichero de configuración, Ldap. Este método lanzará en el navegador del usuario una ventana emergente solicitando las credenciales. Si el usuario introduce de forma errónea por tres veces las credenciales, se le devolverá un error de acceso al sistema. Si estas se han suministrado de forma correcta, procederá a analizar la petición del usuario y su viabilidad según las ACL establecidas.

### 3.2.3 Listas de control de acceso

Una vez el sistema es capaz de identificar a los usuarios que realizan peticiones se debe crear una estructura de listas de acceso ACL que cumpla con la política de accesos restringidos.

En primer lugar se deben definir las ACL en las que se les dotará de un nombre identificativo y el formato del mismo y los elementos que forman parte, estos últimos pueden ser un fichero de texto plano. Por ejemplo:

```
> acl Ips_Sin_Autenticacion src "/opt/squid46/etc/acls/Ips_Sin_Autenticacion.txt"
```

Una vez se han definido las ACLs se debe definir la acción que realizará squid cuando se cumpla la condición de las ACL. Esta configuración se realiza con el comando `http_access allow / deny ACL`. Por ejemplo:

> http\_access allow lps\_Sin\_Autenticacion

El procesamiento que realiza squid es **secuencial** por lo que el orden en el que se establecen los comandos "http\_access" implicarán en el correcto funcionamiento de la política de accesos.

Teniendo en cuenta lo indicado en el párrafo anterior, existen **dos estrategias** a seguir para implementar la estrategia de listas de control de acceso. La primera de ellas se basaría en **habilitar el acceso a todos los contenidos** y bloquear los que no se quieren que se tenga acceso. Esta política es menos restrictiva ya que por defecto, todos los usuarios tienen acceso a la mayoría de contenidos.

La segunda estrategia es inversa a la anterior, se **bloquean todos los contenidos** pero se habilitan los contenidos que pueden ser consultados según los perfiles. En ambos casos, la última directiva "http\_access" es la que concederá o bloqueará el acceso por defecto.

En el caso que ocupa este TFG la estrategia escogida es la mas restrictiva, por la cual se crearán ACLs de acceso para al final bloquear el resto de contenidos.

La estructura de ACLs que se ha planteado es la siguiente:

- ACL Lista Negra Malware. Esta lista de dominios será actualizada de forma automática con el proceso descrito en el siguiente apartado.

```
acl DOM_Malware dstdom_regex "/opt/squid46/etc/acls/DOM_Malware.txt"
acl IP_Malware src "/opt/squid46/etc/acls/IP_Malware.txt"
```

- ACL Lista para Bloqueo General. Esta lista de dominios está destinada a bloqueos que se decida a nivel general.

```
acl Bloqueos dstdom_regex "/opt/squid46/etc/acls/Bloqueos.txt"
```

- ACL para dominios e ip que no pida autenticacion. Si un usuario solicita algun IP o Dominio de estas lista squid no requerirá que el usuario se haya autenticado.

```
acl Dominios_Sin_Autenticacion dstdom_regex
"/opt/squid46/etc/acls/Dominios_Sin_Autenticacion.txt"
acl lps_Sin_Autenticacion src "/opt/squid46/etc/acls/lps_Sin_Autenticacion.txt"
```

- ACL para los equipos que no requieren autenticacion. Si una IP de la lista realiza peticiones al servidor squid no se solicitará autenticacion.

```
acl Usuarios_Sin_Autenticacion src "/opt/squid46/etc/acls/Usuarios_Sin_Autenticacion.txt"
```

- ACL para usuarios con perfil de Libre. Los usuarios en esta lista tendrán permitido el acceso libre de contenidos.

```
acl Usuarios_Libres proxy_auth "/opt/squid46/etc/acls/Usuarios_Libres.txt"
```

- ACL Lista Blanca Ampliada. Una lista adicional de dominios abiertos para un grupo concreto de usuarios.

```
acl Usuarios_LBAmpliada proxy_auth "/opt/squid46/etc/acls/Usuarios_LBAmpliada.txt"  
acl Dominios_LBAmpliada dstdom_regex "/opt/squid46/etc/acls/Dominios_LBAmpliada.txt"
```

- ACL para Lista Restringida de bloqueo de dominios a resto de usuarios.

```
acl Dominios_Restringidos dstdom_regex "/opt/squid46/etc/acls/Dominios_Restringidos.txt"
```

- ACL para la Lista Blanca de Dominios. Todos los dominios que tendrán acceso todos los usuarios.

```
acl Dominios_Blancos dstdom_regex "/opt/squid46/etc/acls/Dominios_Blancos.txt"
```

- ACL para usuarios sin Internet. Los miembros de esta lista tendrán bloqueado el acceso.

```
acl Usuarios_Sin_Internet proxy_auth "/opt/squid46/etc/acls/Usuarios_Sin_Internet.txt"
```

Ademas de las ACLs que se han definido, existen otro tipo de ACL que garantizan el correcto funcionamiento del sistema:

- `acl purge method PURGE`

Para poder eliminar páginas de la cache de squid.

- `acl SSL_ports port 443`

Para poder habilitar los puertos utilizados para HTTPS.

- `acl Safe_ports port 80`

Para poder definir que puertos son seguros.

- `acl CONNECT method CONNECT`

Para poder habilitar el metodo de conexión HTTPS.

A continuación se indica el **orden** de los permisos que se aplican sobre las ACLs antes indicadas. El orden es crítico para garantizar el correcto funcionamiento del sistema.

```
# Denegamos la conexión a puertos no autorizados.  
http_access deny !Safe_ports
```

```
# Denegamos el metodo CONNECT a puertos no definidos como SSL.  
http_access deny CONNECT !SSL_ports
```

```
# Bloqueo de dominios definidos como origen de Malware.
http_access deny DOM_Malware

# Bloqueo de direcciones IP definidos como origen de Malware.
http_access deny IP_Malware

# Bloqueo de la lista de dominios incluidos en la ACL Bloqueos.
http_access deny Bloqueos

# Acceso a los Dominios de la lista sin autentificacion.
http_access allow Dominios_Sin_Autenticacion

# Acceso a las direcciones IP de la lista sin autentificacion.
http_access allow Ips_Sin_Autenticacion

# Acceso a todos los contenidos desde las direcciones IP de usuarios
que no requieran autentificacion.
http_access allow Usuarios_Sin_Autenticacion

# Acceso a todos los contenidos para los usuarios autenticados que
pertenece a la ACL.
http_access allow Usuarios_Libres

# Acceso a los usuarios de la lista LBAmpliada a los dominios de la lista.
http_access allow Dominios_LBAmpliada Usuarios_LBAmpliada

# Bloqueo de acceso a los dominios de la lista
http_access deny Dominios_Restringidos

# Acceso a todos los usuarios autenticados a la lista de
Dominios_Blanco.
http_access allow Dominios_Blanco Usuarios_Autenticados

# Bloqueo de acceso al resto de dominios a los usuarios autenticados.
http_access deny Usuarios_Autenticados all

# Al final denegamos el acceso a todos los otros sitios por este PROXY
http_access deny all
```

Como se ha indicado antes, el orden de análisis de los accesos que realiza squid es secuencial. Por lo tanto, cuando una petición llega al servicio, se le aplicará la primera directiva que se cumpla en el procesamiento, el resto de directivas no serán analizadas.

#### 3.2.4 Lista negra Malware

Una de las principales funciones a implementar en un sistema de filtrado web recae en el **bloqueo** de acceso a dominios que se consideren inseguros.

La gestión de este tipo de listas ha crecido en complejidad a lo largo de tiempo debido al incremento de ataques sobre la seguridad informática de las organizaciones. Esto ha ocasionado que dicha gestión se haya profesionalizado y comercializado. Es posible contratar un **servicio** encargado de generar y mantener diferentes listados de dominios para que nuestros sistemas las utilicen como fuente.

En el caso de este TFG, la implementación de esta funcionalidad también se basará en listados públicos y de acceso gratuito. Esto no invalidaría el hecho de que en un momento dado se pudiese optar por configurar el acceso a un listado contratado.

En este apartado se presentan **dos alternativas** de acceso gratuito. La primera de ellas está basada en una lista generada por el Centro Criptológico Nacional del Gobierno de España (CCN) gracias a su programa CCN-Cert encargado de la defensa frente a las Cyberamenazas [30]. La segunda alternativa se trata de un script creado para este sistema, el cual se nutre de fuentes gratuitas de listas negras.

### Script EVA del CCN-Cert

CCN-Cert ha publicado una guía de configuración segura de servicios proxys llamada “CCN-STIC 660 Seguridad en Proxy” [31].

En dicha guía, en su capítulo 9, presenta un script de filtrado web denominado “EVA”. Este script se nutre de varios orígenes de datos, entre ellos el propio organismo CCN-Cert para generar un listado de dominios y direcciones IP para ser bloqueadas por proxys como por firewalls o sistemas de filtrado antispam. Estas listas se ofrecen por el programa REYES.

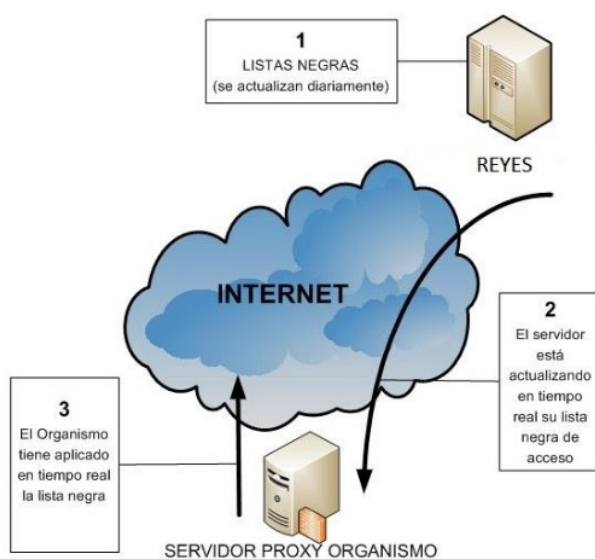


Figura 22: Lista Negra Reyes CCN-Cert

Para poder tener acceso a dicho listado, es necesario solicitar un certificado al propio organismo para realizar la descarga de la lista negra. Debido a que dicho trámite no es posible realizarse a tiempo de la realización de este TFG, se indicará el procedimiento para su configuración.

Como paso inicial se deben instalar los paquetes necesarios del sistema:

```
> apt-get install libconfig-inifiles-perl libjson-perl
```

El siguiente paso es copiar y configurar el script `eva.pl` que facilita el propio CCN-Cert en la guía antes citada.

El script `eva.pl` necesita de un fichero localizado en la misma ruta con el nombre `eva.ini` y que tiene el formato siguiente y que también aparece en la guía:

```
[General]
# Para usar un proxy, descomentar las siguientes lineas (proxy_...) y rellenar los datos:
#proxy=http://10.10.10.10:3128
#proxy_username=user
#proxy_password=12345
# Las listas de url se agrupan en secciones [Listx]
# Si alguna lista requiere uso de certificado x509, con su clave,
# se declaran en la seccion General.
# El formato es Listsx_cert y Listsx_key
# que apuntan a ficheros con el certificado y llave correspondiente
# Si ademas se requiere token (Authorization: Token xxxx),
# el formato es Listsx_token=[token]
Lists1_cert_file=./ccncert.pem
Lists1_key_file=./ccncert.key
Lists1_token=Jniu4JKI6DuW9ruXq/63JuCyeeDjun27Ai/Mjes7eJ0=
timeout=30
# Para descarga automatica de todas las listas, usar "automatic".
# Ejemplo:
# automatic=https://reyes.ccn-cert.cni.es/reyes/api/v1/blacklists
[Lists1]
auto=https://reyes.ccn-cert.cni.es/reyes/api/v1/blacklists
auto_url=https://reyes.ccn-cert.cni.es/reyes/api/v1/blacklist/
[Lists2]
spamhaus_drop=http://www.spamhaus.org/drop/drop.txt
spamhaus_edrop=http://www.spamhaus.org/drop/edrop.txt
#local_list1=file://usr/local/etc/eva_local_list1.txt
```

Como se puede observar es posible configurar varias listas, tanto de orígenes externos como internos.

La salida del script `eva.pl` es un directorio llamado "squid" en el cual existen dos ficheros, "Hosts.txt" e "Ips.txt". El primero de ellos contiene lista de dominios a bloquear y el segundo las direcciones IP.

Estas listas pueden integrarse como ACLs con la siguiente configuración en el fichero `squid.conf`:

```
> acl CCN_IP_list dst -n "/etc/squid/Ips.txt"
> acl CCN_host_list dstdomain -i "/etc/squid/Hosts.txt"
> http_access deny CCN_IP_list
> http_access deny CCN_host_list
```

El CCN-Cert también facilita un script de bash para automatizar la descarga llamado “eva\_run.sh” cuyo código está disponible la guía citada antes. Dicho script puede configurarse como tarea programada en el fichero crontab del sistema.

### **Script actumalware**

Debido a la imposibilidad de implantar el script EVA, se opta por realizar un sencillo script en bash que realice una función similar a EVA y utiliza listas negras gratuitas. El script se ha denominado “actumalware”.

El script se nutre de tres páginas webs de listas malware:

- malwaredomains.com
- www.spamhaus.org
- www.squidblacklist.org

El código del script está disponible en el ANEXO 3. A continuación se describe la ejecución del mismo.

1. El script se descarga las listas de las páginas web, sólo si estas se han modificado.
2. Se tratan las listas para adecuarlas al formato de acl de squid.
3. Se reconfigura squid para que cargue las listas nuevas.

El script genera los ficheros de texto en la ruta que se le indique y que debe referenciarse en las ACL del fichero squid.conf.

Una vez creado solo resta configurar la tarea programada en el fichero /etc/crontab para que se ejecute la actualización con la programación que sea necesaria.

#### **3.2.5 Inspección de tráfico cifrado.**

En la actualidad se ha incrementado la concienciación sobre la seguridad de las comunicaciones dentro de los sistemas web. En el origen, se había restringido el uso de protocolos como HTTPS con SSL o TLS para comunicaciones en entornos de comercio electrónico o donde existiese un tráfico de credenciales de usuario.

Pero esa política ha cambiado, hoy en día, gigantes tecnológicos como *Google* o *Microsoft*, fuerzan a que **todo el tráfico** que se dirija a sus servicios se realice con protocolos seguros.

En lo que respecta al control de tráfico que realiza Squid varía según el protocolo que se utilice. En el caso de HTTP, Squid es capaz de registrar todo el tráfico que se genera entre el cliente y el servidor web de destino. Sobre este tráfico es posible aplicar reglas de acceso y registrar en log todas las comunicaciones realizadas, de tal forma que es posible seguir una traza completa del comportamiento del cliente.

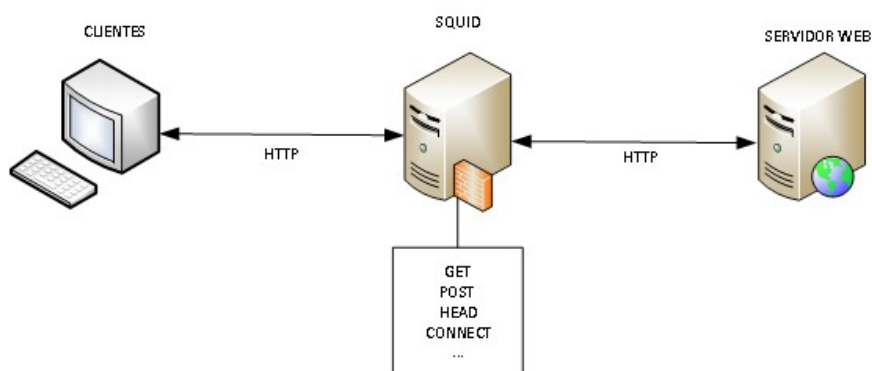


Figure 23: Interacción Cliente Squid Web

Por otro lado, en el caso de tráfico HTTPS, Squid solo es capaz de registrar el tráfico inicial que se establece entre el cliente y el servidor web con el método CONNECT. Todo el tráfico que se genera tras esta conexión es opaco al servidor proxy, por lo tanto este solo es capaz de denegar o permitir el acceso al mismo, pero no registrar la comunicación realizada dentro de dicho dominio al que se ha conectado el cliente.

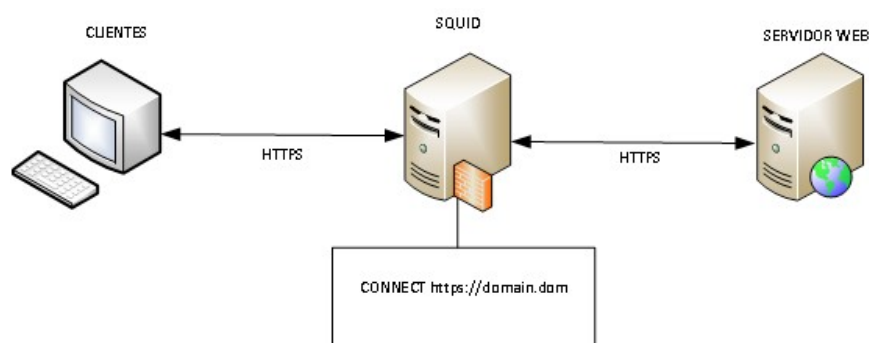


Figure 24: Interacción Cliente Squid Web Cifrada

Ante esta situación, los dispositivos de seguridad han implementado diferentes métodos para poder **inspeccionar este tráfico**. El más común se basa en romper la comunicación segura entre cliente y servidor destino. Esta técnica en la disciplina de la seguridad informática se denomina ataque “*man-in-the-middle*”, por la cual el servidor proxy se transforma en un intermediario entre el cliente y el servidor para el protocolo HTTPS. De tal forma que el servidor proxy es el encargado de establecer las conexiones seguras entre cliente y el destino.



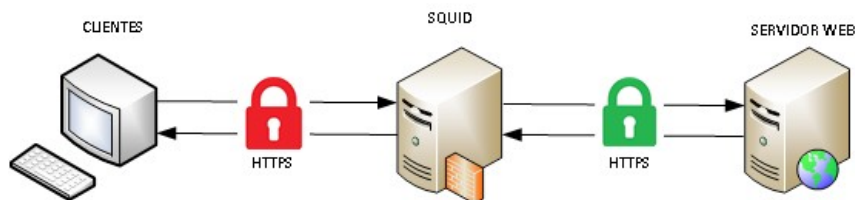


Figura 25: Intercepción SSL

Para poder realizar esta funcionalidad, el servidor Squid debe generar el **par certificado – clave** con el que se realice la comunicación. Este certificado deberá contener las **mismas especificaciones** que tenga el del servidor al que se quiere acceder. Con ellos establecerá la comunicación HTTPS con el cliente y por otro lado establecerá la comunicación con el servidor web destino con los certificados que provea el mismo.

Gracias a lo indicado antes, el servidor es capaz de descifrar el tráfico HTTPS para poder analizar todas las peticiones que se realizan. Una vez realizado puede aplicar las reglas de acceso que tiene establecidas.

A continuación se describen los pasos necesarios para poder implementar esta funcionalidad con el servicio Squid.

- Generación de certificado de los servidores

Como se ha indicado antes, es necesario establecer una comunicación segura entre el cliente y el servidor proxy. Para ello, se debe utilizar un certificado, que en este caso será generado con la herramienta Openssl antes instalada en el servidor.

```
> openssl req -new -newkey rsa:4096 -sha256 -days 1460 -nodes -x509 -extensions v3_ca -keyout proxyCA.pem -out proxyCA.pem
```

La ejecución de este comando nos solicitará datos para ser insertados en el certificado, como el código de país, provincia, localidad, unidad organizativa, FQDN y dirección email de contacto.

Tras la ejecución se genera un fichero proxyCA.pem, el cual se referenciará en la configuración del propio Squid. Antes crearemos una carpeta “ssl\_cert” dentro de la configuración y daremos acceso al usuario que ejecuta squid.

- Para ello debemos incluir la siguiente configuración en el fichero squid.conf:

```
http_port 3128 ssl-bump cert=/etc/squid/ssl_cert/myCA.pem generate-host-certificates=on
dynamic_cert_mem_cache_size=4MB
sslcrtd_program /opt/squid46/libexec/security_file_certgen -s /var/lib/ssl_db -M 4MB
acl step1 at_step SslBump1
```

```
ssl_bump peek step1
ssl_bump bump all
tls_outgoing_options cafile=/usr/local/openssl/cabundle.file
```

En esta configuración se destaca la configuración del parámetro “**ssl-bump**” en el cual se indica a squid que debe utilizar el certificado generado, así como que debe generar certificados de forma dinámica con el parámetro “**generate-host-certificates**” y el tamaño de la memoria cache para los mismos.

Con el parámetro **sslcrttd\_program** [32] se especifica el programa encargado de generar los certificados y la ruta donde se almacenarán.

- Es necesario generar una **cache intermedia** para el almacenamiento de certificados TLS, para ello se ejecuta:

```
> /opt/squid46/libexec/security_file_certgen -c -s /var/lib/ssl_db -M 4MB
```

Cada vez que se modifique el certificado raíz será necesario reinicializar esta base de datos.

Para finalizar se otorga acceso al usuario que ejecuta el servicio de squid:

```
> chown proxy.proxy -R /var/lib/ssl_db
```

- Los navegadores clientes han implementado **métodos de verificación de certificados** para prevenir este tipo de configuraciones, de tal forma que suelen validar la **entidad certificadora** del certificado con el que se establece una comunicación. Si el navegador detecta una anomalía devolverá un mensaje de advertencia al usuario sobre la validez del certificado.

Para evitar este tipo de mensajes es necesario cargar nuestro certificado autogenerado como certificado raíz en los navegadores de los usuarios.

En primer lugar, se transforma el certificado generado al formato “.der” el cual es aceptado por los navegadores para ser importado. Para ello se ejecuta:

```
> openssl x509 -in proxyCA.pem -outform DER -out proxyCA.der
```

Este fichero debe ser importado en los almacenes de certificados de los navegadores de los clientes. En este caso, se detalla el proceso de importación de forma manual, en un navegador Internet Explorer 11.

En el menú de “Opciones de Internet”, dentro de la pestaña de “Contenido” podemos pulsar sobre el botón “Certificados”, el cual

abrirá el menú con las distintas opciones del almacén de certificados. En nuestro caso, se utiliza la pestaña “Entidades de certificación raíz de confianza”. Este almacén es el utilizado por el navegador para validar el certificado emisor. En dicha pestaña está disponible el botón “Importar...”

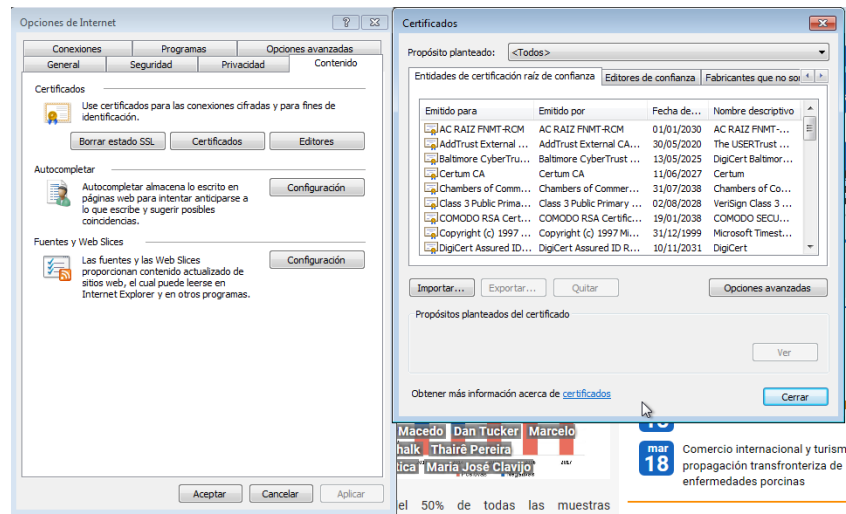


Figura 26: Instalación Certificado Raiz Internet Explorer 11

Una vez pulsado el navegador abre un asistente en el cual solicita información para la importación del certificado como puede verse en las siguientes imágenes:

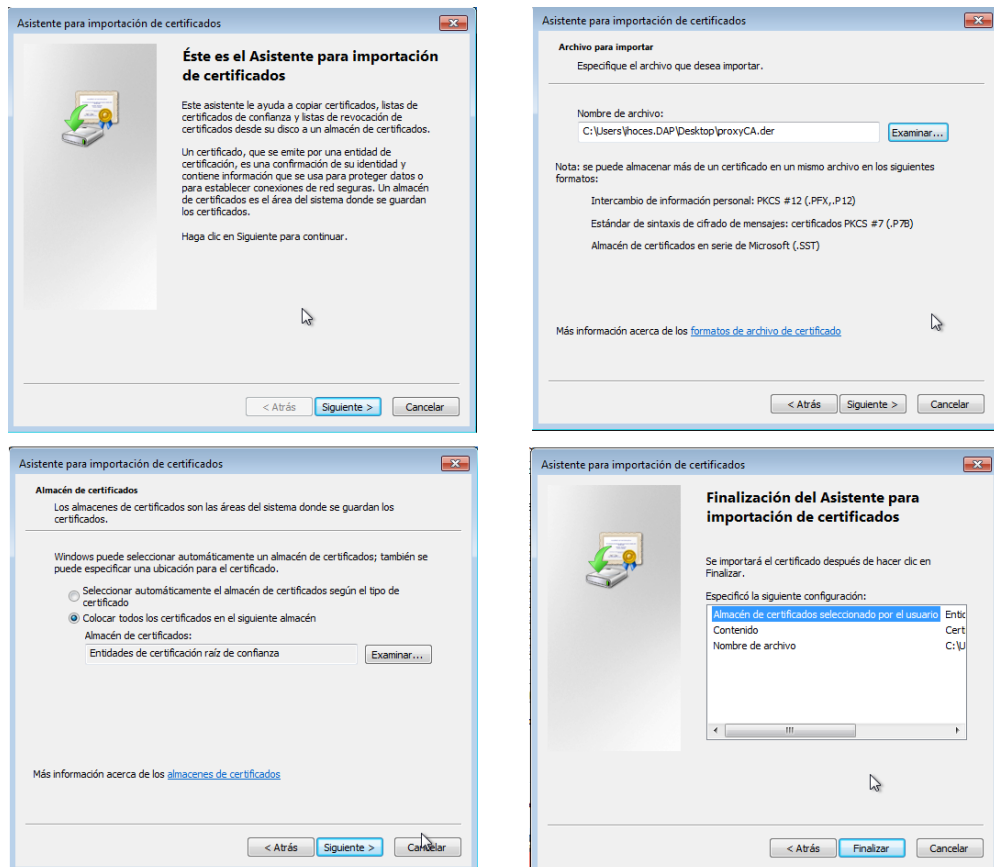


Figura 27: Asistente de Importación de Certificados

Al final mostrará una advertencia sobre la importación del certificado.

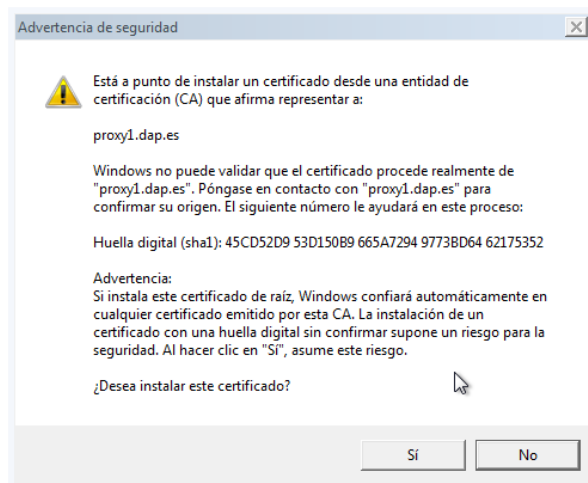


Figura 28: Advertencia de Seguridad Importación de Certificados

Una vez realizada esta configuración podemos revisar el fichero de logs del servicio squid y detectar que aparece tráfico HTTPS mas allá del método CONNECT. Por ejemplo:

```
- 10.239.216.149 - ihoces [19/Mar/2019:16:31:26 +0100] "GET https://maps.gstatic.com/mapfiles/embed/images/entity11.png HTTP/1.1" 200 5321 TCP_MISS:HIER_DIRECT
```

### 3.2.6 Antivirus de Navegación

Una de las funcionalidades que otorgan un nivel superior de seguridad al sistema es el **escaneo de virus durante la navegación**.

Como se indicó en apartados anteriores, la solución a implementar en este apartado se basa en el antivirus open source Clamav. Realizamos su instalación con los siguientes comandos:

```
> apt-get install clamav clamav-daemon
```

Indicamos al sistema que el arranque del servicio se realice de manera automática:

```
> systemctl start clamav-freshclam
```

Una vez instalado el motor de antivirus Clamav procedemos a instalar el modulo de integración con squid, denominado SquidClamav. Este módulo funciona con protocolo abierto icap (Internet Content Adaptation Protocol, RFC 3507 [33]). Este protocolo sirve para redirigir contenidos para realizar filtrado y conversión. Los ejemplos de utilización

son antivirus, traducción dinámica, inserción automática de mensajes, compresión y filtrado de contenidos.

El funcionamiento se basa en el modelo cliente – servidor. El cliente ICAP recibe las peticiones del cliente del servicio y es **derivada** al servidor ICAP, el cual realizará la transformación u operación. Una vez realizada será devuelta al cliente que solicitó el contenido en primer lugar.

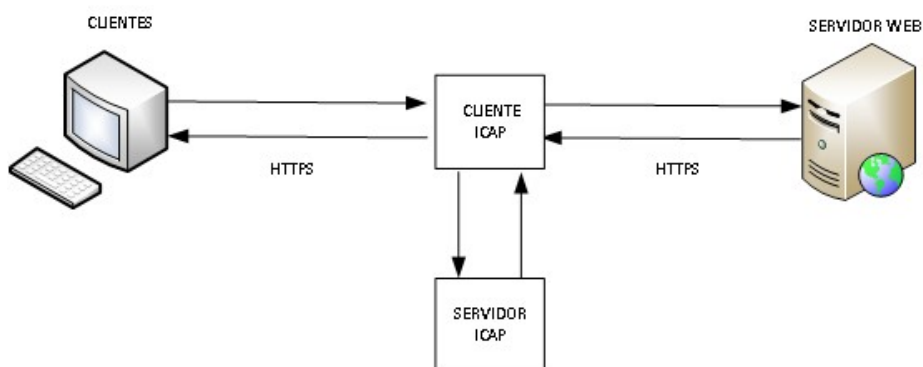


Figura 29: Funcionamiento ICAP

Por lo tanto, es necesario instalar en el servidor proxy además del cliente, el servidor icap. En este caso se instalará el servidor c-icap.

```
> apt-get install curl libcurl4-gnutls-dev c-icap libicapapi-dev
```

Ahora realizamos la instalación del cliente SquidClamav con los parámetros necesarios para que funcione contra un servidor c-icap.

```
> tar zxvf squidclamav-6.16.tar.gz
> cd squidclamav-6.16
> ./configure --with-c-icap
> make
> make install
```

A continuación, realizamos la configuración del servidor c-icap. La carpeta donde se encuentran los ficheros de configuración es “/etc/c-icap”. En dicha carpeta existen dos ficheros que deberemos configurar, “squidclamav.conf” y “c-icap.conf”.

En el primero de ellos podemos modificar la URL a la que redireccionará el antivirus Clamav donde indica que existe un virus. Esto se realiza en el parámetro “redirect”. En nuestro caso:

```
> redirect http://proxy1.dap.es:81/virus.html
```

Otro parámetro a tener en cuenta es clamd\_local, clamd\_ip y clamd\_port. El primero de ellos especifica la ruta al servicio que está corriendo en local en el servidor. Los otros dos parámetros se utilizan para conectar con una IP y Puerto por el que puede estar arrancado el servicio.

En el fichero “c-icap.conf” debemos especificar los siguientes parámetros:

- ServerAdmin usuario@dominio.es

Cuenta de correo electrónico del usuario administrador del sistema.

- ServerName proxy1.dap.es

Identificador del servidor.

- Service squidclamav squidclamav.so

Publicación del servicio de antivirus.

A continuación se especifican alguno de los parámetros que se deberían tener en cuenta para realizar un ajuste de rendimiento del sistema.

- StartServers 3

Número de procesos servidor con los que se iniciará el servicio.

- MaxServers 10

Número máximo de procesos servidor permitidos.

- MinSpareThreads 10 / MaxSpareThreads 20

Cantidad de hilos disponibles por proceso c-icap.

- ThreadsPerChild 10

Número de hilos por proceso hijo.

Ahora solo resta habilitar el servicio c-icap para que se arranque en el inicio del sistema, para lo cual editaremos el fichero “/etc/default/c-icap” donde añadiremos:

```
> START=yes
```

Una vez tenemos el servicio arrancado, debemos configurar Squid para que se integre la funcionalidad de análisis del antivirus. Para ello añadiremos al fichero “squid.conf”:

```
> icap_enable on
> adaptation_send_client_ip on
> adaptation_send_username on
> icap_client_username_header X-Authenticated-User
> icap_service service_req reqmod_precache bypass=1 icap://127.0.0.1:1344/squidclamav
> adaptation_access_service_req allow all
> icap_service service_resp respmod_precache bypass=1 icap://127.0.0.1:1344/squidclamav
```

```
> adaptation_access service_resp allow all
```

### 3.2.7 Personalización de mensajes de error.

El proxy squid mantiene una serie de ficheros html a los que deriva las peticiones que han generado errores de respuesta. Es posible especificar la ruta donde se encuentren esos ficheros en el propio fichero de configuración "squid.conf".

Con el parámetro "error\_directory" especificamos la ruta a la carpeta, en el caso concreto establecemos el valor como "/opt/squid46/etc/errores". Dentro de dicha carpeta se encuentran los **ficheros html** con nombres de error encabezados por "ERR\_". Por ejemplo, "ERR\_ACCESS\_DENIED". Cuando el proxy deniegue el acceso a una página web, enviará al cliente hasta esta página con la descripción del error.

En este caso, se ha utilizado una hoja de estilo que será publicada por el mismo servidor proxy con un servidor http. En este caso utilizaremos un servidor **Apache2** el cual se instala y configura con los comandos siguientes:

- Instalamos el servicio de Apache2.

```
> apt-get install apache2
```

- Configuramos apache2 para que funcione por el puerto 81. Editamos el fichero "/etc/apache2/ports.conf" donde debemos especificar el parámetro "Listen" al puerto 81.

De forma adicional se debe modificar el fichero "/etc/apache2/sites-enabled/000-default.conf" en el que incluimos un VirtualHost para el puerto 81.

- En la ruta del VirtualHost, "/var/www/html", se creará el fichero estilosError.css al cual se hará referencia en las páginas html de error para que lo utilice como hoja de estilo. Por otro lado, se copian las carpetas "images" y "css" necesarios para que funcione la hoja de estilos y las imágenes.
- Habilitamos el arranque automático del servicio y lo inicializamos:

```
> systemctl enable apache2  
> systemctl start apache2
```

A continuación se incluye una imagen de una de las páginas de error personalizadas. En el Anexo 4 se encuentra disponible el código HTML de una de las páginas de error.

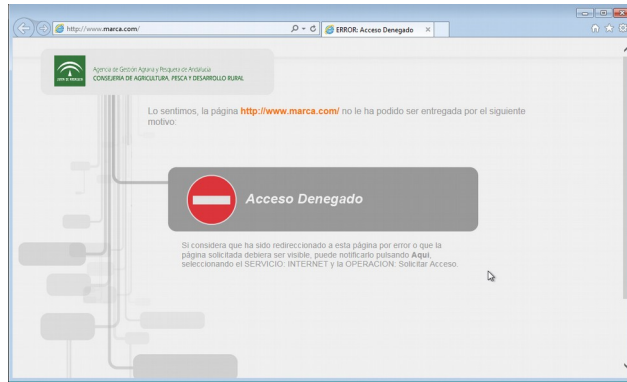


Figura 30: Página de error Acceso Denegado

De la misma forma, se configura la página a la que se redirige en caso de que el análisis de motor de antivirus detecte una amenaza.



Figura 31: Página de Virus Detectado

### 3.3 Configuración del servicio de balanceo de carga

Una vez verificado el correcto funcionamiento de un nodo de servicio proxy, es necesario implementar las configuraciones necesarias para que el sistema sea **escalable**.

Existen dos estrategias de escalado, el **horizontal** y el **vertical**. El escalado vertical se basa en dotar de mayor capacidad al nodo que ejecuta el servicio. En el caso que nos ocupa, dotar de mayor capacidad de cómputo y memoria al proxy para poder soportar mayor número de conexiones simultáneas y que su velocidad de respuesta a las peticiones sea rápida.



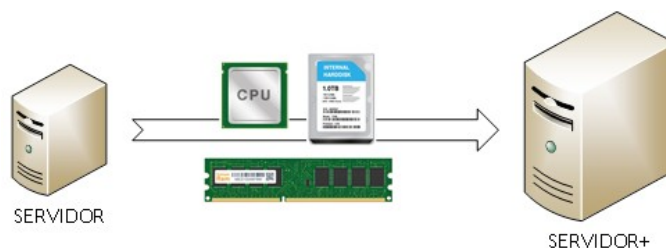


Figura 32: Escalado Vertical

La segunda estrategia es el escalado horizontal. En este caso, se alcanza la escalabilidad con el reparto de las peticiones entre varios nodos. En nuestro caso, dotar de varios servidores proxy para que las solicitudes de los clientes sean derivadas a unos u otros indistintamente.

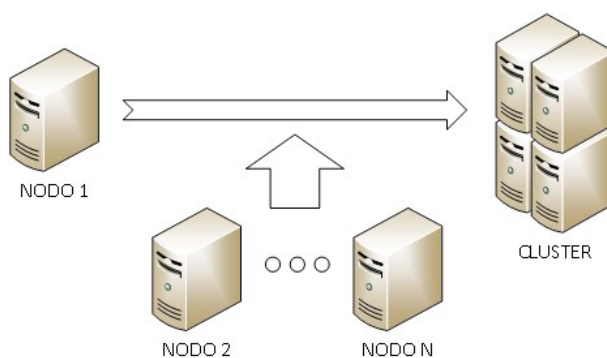


Figura 33: Escalado Horizontal

El escalado horizontal otorga una ventaja adicional y es la capacidad de dotar de **alta disponibilidad** al sistema. Al introducir un elemento que se encarga de derivar las peticiones a los distintos nodos puede incluirse un sistema de verificación de estado de los nodos. De esta forma, en caso de que un nodo deje de responder, no se le deriven peticiones de los clientes.

Para realizar esta funcionalidad se introduce un sistema de **balanceo de carga** (Load Balancer) que será realizado por el servicio Haproxy. Tal y como se indicó antes, la elección de esta herramienta viene determinada por encontrarse ya implantada en la infraestructura donde se instalan los componentes del sistema de navegación. Con esta medida se mantiene la homogeneidad en las herramientas que se utilizan en este entorno.

En cualquier caso, la funcionalidad requerida para este sistema puede ser cubierta por multitud de sistemas de balanceo de carga.

### 3.3.1 Instalación de un segundo nodo proxy

Gracias a la funcionalidad que otorga el entorno utilizado para este TFG, **Vmware Vsphere 6.5**, podemos realizar un “clon” del nodo squid. Con esta nueva máquina virtual sólo debemos aplicar los

**cambios necesarios** para que no provoque problemas con el nodo inicial.

El listado de cambios que se realiza es el siguiente:

- Cambio de nombre de servidor a proxy2.
- Cambio de dirección IP para el nuevo servidor.
- Añadir en DNS el registro A y PTR del nuevo servidor.
- Modificación de fichero squid.conf parametro visible\_hostname.
- Unión a dominio y generación de fichero PROXY.keytab con el comando:

```
> msktutil -c -b "OU=SRV_LINUX" -s HTTP/proxy2.dap.es -k /opt/squid46/etc/PROXY.keytab --
computer-name PROXY2-K --upn HTTP/proxy2.dap.es --server dc1.dap.es --verbose --enctypes
28
```

- Modificación de permisos de acceso sobre el fichero PROXY.keytab.
- Modificación de referencias a “proxy1.dap.es” en páginas de error por “proxy2.dap.es”.
- El certificado que utilizará el servidor proxy2 será el mismo que el que se ha generado en la instalación del nodo proxy1. Al estar implantado en los clientes no sería necesario generar uno para cada nodo. En ese caso, sería necesario instalar todos los certificados en los equipos clientes.
- Una vez realizados estos cambios se verifica que el nodo proxy2 es operativo como servidor proxy.

### 3.3.2 Instalación del servicio de balanceo de carga

Para ofrecer el servicio de balanceo de carga entre los dos nodos proxy se opta por implantar un sistema en alta disponibilidad. Con esta medida se evita que el servicio de balanceo de carga pase a ser un punto único de fallo para el sistema de navegación.

Para ello se dota al sistema de dos nodos Haproxy con configuraciones de balanceo idénticas que compartirán una IP de frontend con el sistema keepalived. Este servicio está implementado con el protocolo **VRRP** (Virtual Router Redundancy Protocol [34]).

Este sistema establece roles de MAESTRO y BACKUP a cada uno de los nodos Haproxy. De esta manera, el servidor que ostenta el

**rol de MAESTRO** será el encargado de responder a las peticiones realizadas a la IP de frontend.

El servidor con **rol de BACKUP** se mantendrá a la espera. En caso de que el servicio keepalived detecte que el servidor MAESTRO no se encuentra operativo, cambiará el rol al servidor de BACKUP para que pase a ser MAESTRO.

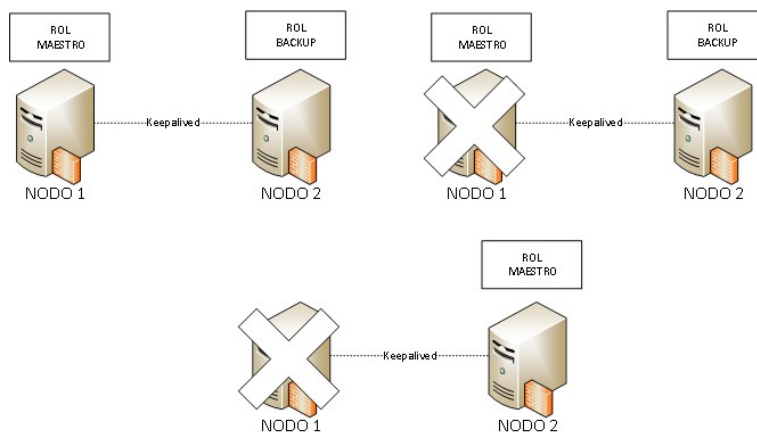


Figura 34: Cambio de roles - KeepAlived

En caso de que el antiguo nodo MAESTRO vuelva a estar operativo, será el propio sistema el que devuelva los roles a sus propietarios originales. Esta configuración se realiza con pesos que se asigna a cada nodo, de tal forma que el nodo MAESTRO será el que mayor peso se le conceda.

En el ANEXO 5 se incluye la configuración realizada en el servicio keepalived en el fichero de configuración “/etc/keepalived/keepalived.conf”. A continuación se describe alguno de los parámetros de mayor relevancia en el mismo:

- notification\_email

Cuenta de correo electrónico a la que se le notificarán los cambios de rol en el sistema.

- smtp\_server  
Pasarela de correo electrónico que se utilizará para el envío de correos.
- vrrp\_script

Configuración del script de chequeo de estado de los nodos. En esta definición se establece tanto el método de verificación como la periodicidad o el número de intentos del mismo para que se considere fallido el nodo.

- vrrp\_instance

Definición de la instancia compartida entre los nodos y la configuración del propio nodo dentro de esta.

- state

Definición del rol MASTER o BACKUP.

- interface

Tarjeta de red por la que se prestará el servicio.

- virtual\_router\_id

Identificador utilizado para diferenciar distintas instancia VRRP.

- priority

Peso que se le asigna al nodo que se configura.

- authentication

Configuración de autenticación entre los nodos participes en keepalived.

- virtual\_ipaddress

Dirección IP que se compartirá entre ambos nodos.

Una vez configurado en ambos servidores se verifica que el nodo MAESTRO comienza a responder por la dirección IP virtual indicada en la configuración. En el caso de nuestro sistema se ha establece como 10.239.212.50 la IP virtual.

Para realizar el balanceo de carga se configura el servicio con el fichero de configuración “/etc/haproxy/haproxy.cfg”. Dicho fichero se encuentra estructurado en varias secciones. Las secciones principales son “global” y “defaults”. En cada una de ellas se configuran aspectos diferentes para el comportamiento del servicio.

En el apartado global se establecen configuraciones generales del sistema como la configuración de logs, las estadísticas, el usuario y grupo que ejecutará el servicio o el sistema de certificados que podrá utilizarse.

En el apartado defaults, se realiza la configuración del comportamiento por defecto del sistema, por ejemplo si se requieren páginas de error o los tiempos de timeout del servicio.

En las siguientes secciones es donde se establece la configuración del servicio de balanceo específico para el tráfico que queremos que reciban los nodos proxy squid.

La configuración que se establece se indica a continuación y después se describen cada uno de los elementos más destacables de la misma.

```
listen squid
bind *:3128
mode http
option httplog
balance source
hash-type consistent
option httpclose
cookie SERVERID insert indirect nocache
option forwardfor
server proxy1squid 10.239.212.51:3128 check inter 2000 rise 2 fall 5
server proxy2squid 10.239.212.52:3128 check inter 2000 rise 2 fall 5
```

- bind

Especificamos la IP y el **puerto por el que escuchará** el servicio Haproxy. En este caso el puerto es el mismo por el que escuchan los servidores squid y aceptará las peticiones entrantes por cualquier IP que tenga el servidor.

- mode

Establece el **modo de funcionamiento**. Este parametro acepta tres configuraciones, tcp, para funcionar en modo TCP puro, http , para que el sistema funcione en modo HTTP y pueda analizar las peticiones y health en la cual el sistema sólo responde con un "OK" ante las peticiones.

- balance

**Algoritmo** utilizado para balancear las peticiones. Entre los distintos algoritmos que permite el sistema hace mención a alguno de ellos, roundrobin, source y leastconn. El primero de ellos realiza reparte las peticiones con turnos y según el peso que se le otorgue a cada nodo backend. El segundo de ellos, se realiza un hash de la dirección IP del cliente y se divide entre el peso total de los servidores para designar cual atenderá la petición. El tercero de ellos evalúa el número de peticiones que reciben los nodos y la entregará al que menos tenga en ese momento.

- option httpclose

Habilitamos la capacidad para poder cerrar la conexión una vez se ha transferido.

- option forwardfor

Habilitamos a Haproxy para que incluya la IP del cliente al transferirla al servidor backend. Este parámetro es esencial para poder **identificar en los squid la IP del cliente** y no la del balanceador de carga.

- server

Declaración de un nodo backend de respuesta. En este caso los servidores squid que procesarán las peticiones de los clientes.

Tal y como se realiza la configuración, el servicio de balanceo de carga recibirá las peticiones a través de la dirección IP virtual establecida por el **puerto 3128**. Dichas peticiones serán derivadas según el algoritmo escogido, hacia los dos nodos proxy que ejecutan el servicio squid.

La elección del algoritmo de balanceo de carga viene establecida ya que, para la comunicación entre un cliente y una página web sea correcta, debe **permitir la persistencia** de la misma. Si no se mantiene o se utiliza un algoritmo que no la soporte, pueden ocurrir errores, como por ejemplo, que el usuario no pueda autenticarse en una página o que se le requiera en varias ocasiones al establecerse desde distintos servidores proxys.

En este caso, además de utilizar un algoritmo que establece una vinculación entre el cliente y el servidor según su IP, se inserta una cookie con la id del servidor de respuesta.

En este punto de la implantación, el sistema ya es capaz de balancear las peticiones entre los nodos proxys squid. Pero esta configuración provoca un fallo en uno de los sistemas de autenticación.

El sistema basado en Kerberos se ha configurado de tal forma que las peticiones son aceptadas desde los nodos proxy con su cuenta de servicio "HTTP\proxy1.dap.es" y "HTTP\proxy2.dap.es". Por el contrario, el sistema de autenticación basado en LDAP funciona sin problemas, solicita con una ventana emergente, las credenciales al usuario.

Al utilizar la dirección IP virtual, la petición no es recibida por alguna de las dos cuentas antes citadas. Ante esto debemos habilitar al sistema para que pueda autenticar desde una nueva cuenta de servicio. Para ello vamos a crear una cuenta de servicio que vincularemos con un registro DNS que apunte a la IP virtual. Dicha cuenta será agregada al fichero PROXY.keytab que se generó en el apartado sobre autenticación Kerberos y que es utilizado por los squid para autenticar con este protocolo.

El procedimiento es el siguiente:

- Crear en los DNS un registro A que apunte el nombre a la IP virtual.
- Creación de usuario en el Active Directory con nombre “HTTP/inet.agenciaagrariaypesquera.es” y nombre NT/2000 “HTTP\_inet”.
- A continuación, se debe crear el SPN y vincularlo a la cuenta de usuario creada, para ello, desde una línea de comandos de un equipo con privilegios de administrador del dominio de Active Directory ejecutamos:

```
> setspn -a HTTP/inet.agenciaagrariaypesquera.es HTTP_inet
```

```
> setspn -a HTTP/inet HTTP_inet
```

```

Administrador: Símbolo del sistema
Microsoft Windows [Versión 6.3.9600]
(c) 2013 Microsoft Corporation. Todos los derechos reservados.
C:\Users\Administrador.DAP>setspn -a HTTP/inet.agenciaagrariaypesquera.es HTTP_inet
Comprobando el dominio DC=dap,DC=es
Registrando valores de ServicePrincipalName para CN=SPN_INET_INET,OU=SERVIDORES,DC=dap,DC=es
    HTTP/inet.agenciaagrariaypesquera.es
Objeto actualizado
C:\Users\Administrador.DAP>setspn -a HTTP/inet HTTP_inet
Comprobando el dominio DC=dap,DC=es
Registrando valores de ServicePrincipalName para CN=SPN_INET_INET,OU=SERVIDORES,DC=dap,DC=es
    HTTP/inet
Objeto actualizado
C:\Users\Administrador.DAP>_

```

Figura 35: Creación de SPN desde línea de comandos Windows.

- Ahora debemos modificar los ficheros PROXY.keytab en ambos nodos squid e insertar las credenciales de esta SPN. Para ello ejecutamos:

```
> kinit HTTP/inet.agenciaagrariaypesquera.es
```

Introducimos la contraseña del usuario que hemos creado y vinculado al SPN.

- Utilizamos la herramienta para modificar el fichero keytab.
- Ktutil
- Añadimos las credenciales de la cuenta y el cifrado utilizado por Kerberos [35](rfc4757).

```
> ktutil: addent -password -p HTTP/inet.agenciaagrariaypesquera.es -k 2 -e rc4-hmac
```

```
> Password for HTTP/inet.agenciaagrariaypesquera.es@DAP.ES:
```

- Repetimos la instrucción, suprimimos el dominio por si la petición es solicitada sin este.

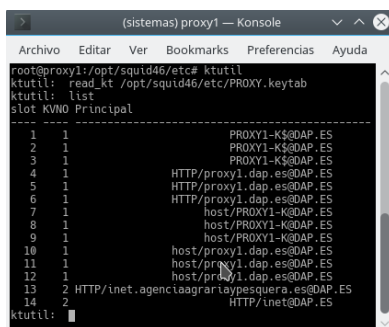
```
> ktutil: addent -password -p HTTP/inet -k 2 -e rc4-hmac
> Password for HTTP/inet@DAP.ES:
```

- Guardamos los cambios en el fichero PROXY.keytab de squid.

```
> ktutil: wkt /opt/squid46/etc/PROXY.keytab
> ktutil: quit
```

En el caso de querer listar el listado de “keys” que contiene el fichero PROXY.keytab podemos hacerlo con los comandos siguientes:

- ktutil
- read\_kt /opt/squid46/etc/PROXY.keytab
- list



```
(sistemas) proxy1 — Konsole
Archivo Editar Ver Bookmarks Preferencias Ayuda
root@proxy1:/opt/squid46/etc# ktutil
ktutil: read_kt /opt/squid46/etc/PROXY.keytab
ktutil: list
slot KVNO Principal
-----
1 1 PROXY1-K@DAP.ES
2 1 PROXY1-K@DAP.ES
3 1 PROXY1-K@DAP.ES
4 1 HTTP/proxy1.dap.es@DAP.ES
5 1 HTTP/proxy1.dap.es@DAP.ES
6 1 HTTP/proxy1.dap.es@DAP.ES
7 1 host/PROXY1-K@DAP.ES
8 1 host/PROXY1-K@DAP.ES
9 1 host/PROXY1-K@DAP.ES
10 1 host/proxy1.dap.es@DAP.ES
11 1 host/proxy1.dap.es@DAP.ES
12 1 host/proxy1.dap.es@DAP.ES
13 2 HTTP/inet.agenciaagrariaypesquera.es@DAP.ES
14 2 HTTP/inet@DAP.ES
ktutil: █
```

Figura 36: Contenido del fichero PROXY.keytab

En este punto podemos configurar en el navegador como servidor proxy “inet.agenciaagrariaypesquera.es” y puerto “3128”. Al realizar las peticiones el sistema será capaz de identificar al usuario con Kerberos.

En caso de querer verificar qué método de autenticación está operativo el sistema podemos analizarlo en el fichero “/var/log/cache.log” donde aparecerán mensajes como:

```
2019/03/21 18:14:34| squid_kerb_auth: User ihoces authenticated
```



## 4. Centralización de Logs.

### 4.1 Descripción del servicios

Los sistemas de información formados por múltiples componentes generan **grandes volúmenes de datos** con sus registros de logs para cada uno de ellos. Algunos de estos componentes pueden generar muchos logs que sean de gran utilidad y por el contrario, otros generar pocos y de poca utilidad.

Los principales motivos que pueden llevar a implementar un sistema que recolecte todos los logs de los componentes son variados, pero los mas importantes son la **resolución de problemas** y la obtención de **conocimientos valiosos** para las organizaciones.

A la hora de resolver problemas, implica la consulta de los distintos logs de cada componente, incluso puede ser necesario el cruce de los mismos a fin de determinar la causa principal de los problemas. Todo ello relacionado por las distintas dependencias que presentan los componentes de los sistemas de información.

La otra utilidad de este servicio es la extracción de información gracias a los datos generados por los logs. Este sistema se convierte en un punto central de fuentes de información diversas donde pueden ser explotadas para generar conocimiento que derivar en nuevas opciones de negocio o mejoras sobre procesos o productos.

En este caso, el sistema elegido para este servicio es el *bundle* de programas desarrollado por la empresa Elastic [36] denominado Elastic Stack. Este sistema está compuesto por los siguientes componentes:

- Elasticsearch
- Logstash
- Beats
- Kibana

El primero de los elementos es Elasticsearch [24]. Este sistema es el encargado de **almacenar los datos, analizarlos y realizar búsquedas** sobre los mismos. Está basado en JSON y utiliza un modelo distribuido y escalable de forma horizontal.

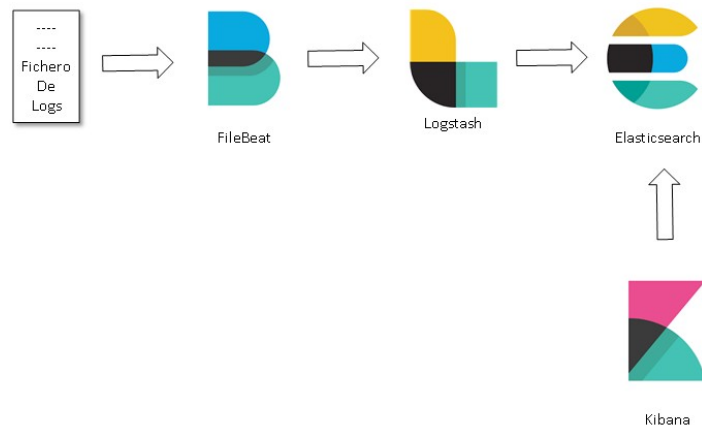
El segundo de los elementos es Logstash [26]. Este componente es el encargado de **centralizar y transformar los logs** que son enviados a Elasticsearch. Funciona con configuraciones pipeline que

tratan datos de múltiples fuentes antes de enviarlas al sistema de almacenamiento.

El tercer elemento es Beats [37]. En este caso, se trata del cliente que se instalará en las diferentes fuentes de información para **enviar los datos** hacia Logstash. Este sistema tiene distintos módulos, cada uno orientado a distintos tipos de fuentes de logs.

El último elemento es Kibana [27]. Este componente es el encargado de la **visualización de los datos** que mantiene Elasticsearch. Entre sus funciones está la ejecución de consultas, creación de visualizaciones personalizadas y paneles de datos que permitan conocer datos importantes en una sola pantalla.

El funcionamiento global del sistema se puede consultar en la siguiente imagen:



*Figura 37: Interacción ELK Stack*

En el caso particular que atañe a este TFG, el funcionamiento del sistema de recopilación de logs seguirá la disposición siguiente:

- Se dispone de un servidor denominado ELKSERVER que tendrá instalado los componentes Elasticsearch, Logstash y Kibana.
- El componente Logstash se configurará para recibir los logs para ser tratados y derivados a Elasticsearch para ser almacenados.
- Kibana se instala y configura para conectar contra el componente Elasticsearch para poder realizar las consultas necesarias.
- Se instala en los servidores proxy, los clientes FileBeat (componente para el traslado de ficheros perteneciente al componente Beats), encargados de trasladar los ficheros que

genere el servicio Squid instalado en los mismos hacia el componente Logstash.

El funcionamiento del sistema dentro de la infraestructura instalada queda reflejada en el siguiente diagrama.

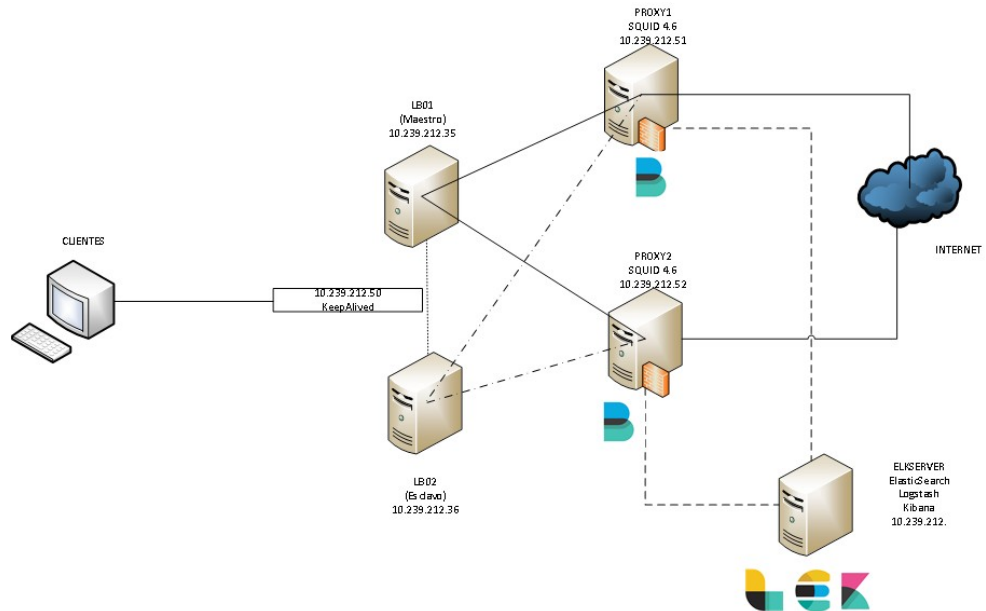


Figura 38: ELK Stack en los elementos

En los siguientes apartados se detallará la configuración de los distintos elementos indicados en los apartados anteriores. De forma adicional se especificará la configuración de los logs en los nodos proxy y sus implicaciones con el sistema de visualización Kibana.

## 4.2 Traspaso de los ficheros de logs

### 4.2.1 Instalación de Elasticsearch

El primer componente que se instala en el servidor ELKSERVER es el sistema de almacenamiento y búsqueda Elasticsearch.

Este sistema tiene como requisito la instalación de una máquina virtual de Java versión 1.8 para ello se ejecuta el comando:

```
> apt-get install default-jre
```

Una vez cumplido dicho requisito se procede a descargar el paquete instalable para el sistema operativo del servidor desde la web del fabricante del producto [38].

Para realizar la instalación se ejecuta:

```
> dpkg -i elasticsearch-6.7.0.deb
```

Los ficheros de configuración del producto se encuentra en la ruta “/etc/elasticsearch” y en ella se encuentra el fichero de configuración “elasticsearch.yml”. Sobre este fichero se realiza la modificación necesaria para que permita consultas desde otros servidores distintos a si mismo:

```
> #AÑADIDO PARA ACCESO EXTERNO
> network.bind_host: 0.0.0.0
> node.master: true
> node.data: true
> transport.host: localhost
```

Habilitamos el servicio para su arranque automático y lo arrancamos por primera vez:

```
> systemctl enable elasticsearch.service
> systemctl start elasticsearch.service
```

Para verificar que el sistema se encuentra activo y en servicio, basta con acceder por http al puerto 9200 del servidor. La respuesta será como la imagen siguiente.



Figura 39: Página Web de respuesta Elasticsearch

#### 4.2.2 Instalación de Logstash

El segundo componente necesario del sistema es Logstash. Este sistema es el encargado de recibir los ficheros de logs y aplicarle reglas para transformarlos antes de enviarlos al componente instalado en el apartado anterior.

Al igual que el componente anterior, este se descarga desde la misma dirección web. Se selecciona el producto y la versión del sistema operativo.

Para realizar la instalación se ejecuta:

```
> dpkg -i logstash-6.7.0.deb
```

El directorio donde se almacenan los ficheros de configuración necesarios del sistema se encuentra en la ruta “/etc/logstash/conf.d”. Dentro de esta carpeta se crean los ficheros con extensión “.conf” que albergarán los parámetros necesarios para el funcionamiento de este componente.

En este caso se realiza la configuración en un único fichero, aunque es posible en dividirlo en varios para separar cada una de las funcionalidades que se describen a continuación.

El fichero de configuración se estructura en tres apartados distintos, cada uno de ellos con una funcionalidad definida. Para más información se puede consultar la documentación oficial del fabricante [39]. La estructura es la siguiente:

- input
- filter
- output

El primero de ellos, input, es el parámetro donde se establece la configuración de logstash referente a la **entrada de datos**. En este apartado, el parámetro principal es el puerto por el que recibirá la información el servicio de logstash. En el caso de este sistema indicamos el puerto 5000 con la configuración siguiente:

```
tcp {  
  port => 5000  
  type => "squid-access"  
}
```

El segundo de los apartados es filter. En este apartado es donde se pueden aplicar filtros a la entrada de datos con el fin de adaptarlos a un formato que pueda ser interpretado de manera correcta por el motor de elasticsearch.

Para realizar el filtrado se hace uso de **patrones grok**(grok pattern). Estos patrones facilitan la traducción de un log a un conjunto de campos diferenciados.

Los patrones grok siguen la forma “%{PATRON:nombre\_del\_campo}”.

Por ejemplo, dado una línea de log como la siguiente:

“Total de peticiones = 2568”

Podemos aplicar el siguiente patrón grok para obtener el valor numérico que es el que tiene importancia en la línea de log.

“Total de peticiones = %{NUMBER:total\_peticiones}”

El tercer apartado corresponde con output. En este apartado se establece la **salida de los datos** ya filtrados hacia el repositorio almacén elasticsearch. Se puede indicar que esos datos deben ser almacenados en un índice concreto dentro del elasticsearch.

La configuración mínima necesaria para el sistema que se implanta sería la siguiente:

```
output {
  elasticsearch {
    hosts => ["elkserver.dap.es:9200"]
    index => "squid-access"
  }
}
```

En ella se indica la conexión con el servicio de elasticsearch y el nombre del índice donde serán referenciados los datos provenientes de los ficheros de logs.

La capacidad de configuración de logstash es muy amplia y excede del objetivo de este TFG, por lo tanto, sólo se indican las configuraciones necesarias para el correcto funcionamiento del sistema objeto del mismo.

El fichero de configuración del componente logstash se encuentra disponible en el ANEXO 6.

#### 4.2.3 Instalación de Beats

Para poder enviar los ficheros de logs desde los servidores proxy squid hacia el repositorio centralizado es necesario instalar el servicio de beats, de forma mas concreta, el módulo FileBeat. Este módulo, perteneciente al ecosistema de Beats, se encargará de tomar los ficheros de logs y enviarlos hacia el sistema logstash.

El cliente Filebeat puede descargarse desde la misma página utilizada para los componentes antes descritos.

Para realizar la instalación se ejecuta el siguiente comando:

```
> dpkg -i filebeat-6.7.0-amd64.deb
```

El fichero de configuración se encuentra en la ruta “/etc/filebeat/” y se denomina “filebeat.yml”. Para configurar el envío de los logs, es necesario modificar los parámetros siguientes que indican los ficheros a enviar y el destino:

```
filebeat.inputs:
enabled: true
paths:
  - /var/log/squid46/access.log

output.logstash:
  # The Logstash hosts
```

```
hosts: ["elkserver.dap.es:5000"]
```

Una vez queda configurado el servicio tomará el fichero y enviará su contenido al servicio de Logstash.

#### 4.2.4 Instalación de Kibana

El último elemento a instalar el es que permitirá la visualización y explotación de los datos recopilados en Elasticsearch. Este componente es Kibana y su instalación se realiza de manera similar a los componentes anteriores.

Ejecutamos el siguiente comando para instalar el componente:

```
> dpkg -i kibana-6.7.0-amd64.deb
```

El fichero de configuración de Kibana se encuentra en la carpeta "/etc/kibana". El fichero de configuración general del servicio se denomina "kibana.yml".

Es necesario realizar las modificaciones siguientes en el fichero de configuración:

- Para establecer la conexión de Kibana a Elasticsearch:  
elasticsearch.hosts: "<http://elkserver.dap.es:9200>"
- Para permitir conexiones a Kibana desde cualquier dirección:  
server.host: "0.0.0.0"
- Para que se ofrezca con un contexto /kibana  
server.basePath: "/kibana"  
server.rewriteBasePath: true

Una vez arrancado el servicio de Kibana podemos acceder a la URL:

```
http://elkserver.dap.es:5601/kibana
```

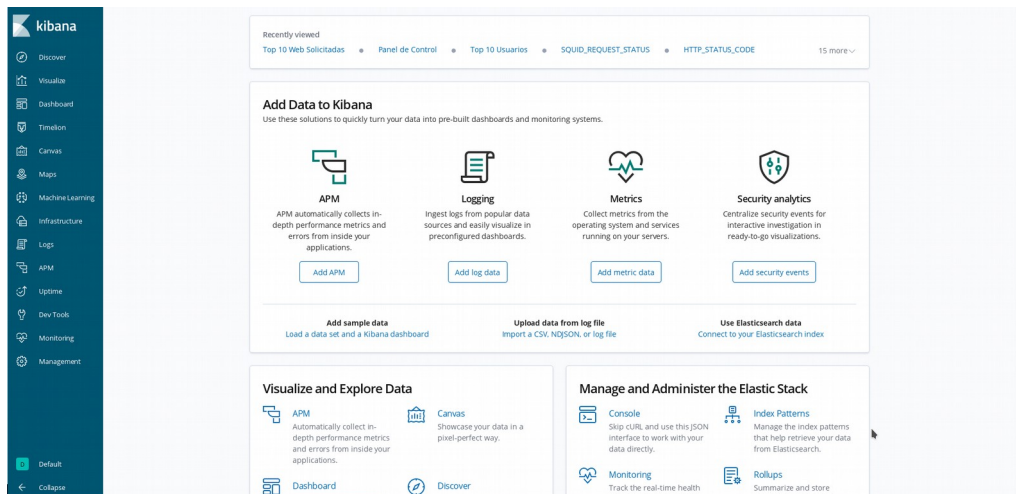


Figura 40: Página principal de Kibana

#### 4.2.5 Formato de logs de Squid

En este punto, el sistema de recopilación de logs ya se encuentra en funcionamiento. Los nodos proxy envían los logs hacia Logstash y este hacia el Elasticsearch. Si accedemos a Kibana podemos consultar los logs y realizar algunas búsquedas. Pero para obtener un mayor provecho a este sistema es necesario realizar una **optimización** de los logs.

En primer lugar, es necesario definir el **formato de log** que se generará en los servidores squid. En el fichero de configuración se puede modificar el formato con el que se presentan los logs con el uso del parámetro “logformat”. El parámetro por defecto del formato de logs de squid es el siguiente:

```
logformat squid %>a %ui %un [%tl] "%rm %ru HTTP/%rv" %>Hs %<st %Ss:%Sh
```

El parámetro anterior denomina al formato como “squid” y a continuación aparecen los distintos elementos que formarán parte de la línea de logs.

En la documentación oficial del software squid-cache [40] puede consultarse en más detalle cada uno de los elementos que pueden configurarse.

En lo que respecta al sistema objeto de esta documentación, la correcta configuración del componente Logstash depende del formato escogido en la configuración de los nodos proxy. Es necesario crear un **patrón grok** que encaje a la perfección con el formato que se utilice, a fin de que se puedan capturar los distintos campos con información relevante.

En este caso utilizaremos una **configuración personalizada**, distinta de la que utiliza por defecto squid, para mostrar un ejemplo de filtrado con patrón grok de logstash.



El formato de log de squid elegido es el siguiente:

```
logformat squid %{X-Forwarded-For}>h %>a %ui %un [%t] "%rm %ru HTTP/%rv" %>Hs %<st %Ss:%Sh
```

Una de las diferencias frente al patrón por defecto, es la inclusión de la **dirección IP** encadenada de saltos, con el parámetro **X-Forwarded-For**. Gracias a esto podremos identificar las direcciones IP de los equipos cliente ya que de lo contrario sólo se mostraría la dirección IP del Haproxy, que es el elemento que deriva la petición a squid.

Una vez modificamos el formato de squid, es necesario incluir el filtro en el fichero de configuración de logstash. Para ello incluimos la siguiente configuración:

```
filter {
  grok {
    match => [
      "message", "%{NOTSPACE:xforw_ip} %{IPV4:src_ip} %{NOTSPACE:id_user} %{NOTSPACE:user} [%{
      {HTTPDATE:logdate}] \\"%{WORD:http_method}          %{URI:request_url}          HTTP/%
      {NOTSPACE:protocol_version}\" %{NUMBER:http_status_code} %{NUMBER:reply_size_include_header} %
      {WORD:squid_status}:%{WORD:squid_hierarch}",
      "message", "%{NOTSPACE:xforw_ip} %{IPV4:src_ip} %{NOTSPACE:id_user} %{NOTSPACE:user} [%{
      {HTTPDATE:logdate}] \\"%{WORD:http_method}          %{URIHOST:request_url}          HTTP/%
      {NOTSPACE:protocol_version}\" %{NUMBER:http_status_code} %{NUMBER:reply_size_include_header} %
      {WORD:squid_status}:%{WORD:squid_hierarch}"
    ]
  }
}
```

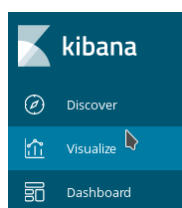
Dentro del apartado grok se incluyen **varios patrones** distintos ya que las líneas que contienen el fichero de logs de squid presentan varios tipos distintos. Es posible añadir distintos patrones para que el sistema pueda identificar las **distintas líneas** del fichero.

Una vez aplicado el cambio y reiniciado el servicio, es necesario verificar que en Kibana se empiezan a registrar los distintos campos definidos en el grok. Es posible que estos se identifiquen pero que no se muestren como campos del índice. Si esto ocurriese, Kibana sería incapaz de utilizar dichos campos para realizar búsquedas o filtros. Para solventarlo hay que indicar a Kibana que realice una recarga de los campos del índice de Elasticsearch.

## 4.3 Visualizaciones y Dashboard

El siguiente paso para aprovechar las capacidades que aporta un sistema como el implantado es la creación de visualizaciones y consultas sobre los datos recopilados. En este apartado se muestran la creación de un juego de **visualizaciones**, sistema como denomina Kibana a sus consultas personalizadas, así como la creación de un panel o *dashboard* formado por un conjunto de visualizaciones.

Dentro de la aplicación Kibana podemos acceder al menú dedicado a las visualizaciones con el acceso en el menú izquierdo en el apartado “Visualize”.



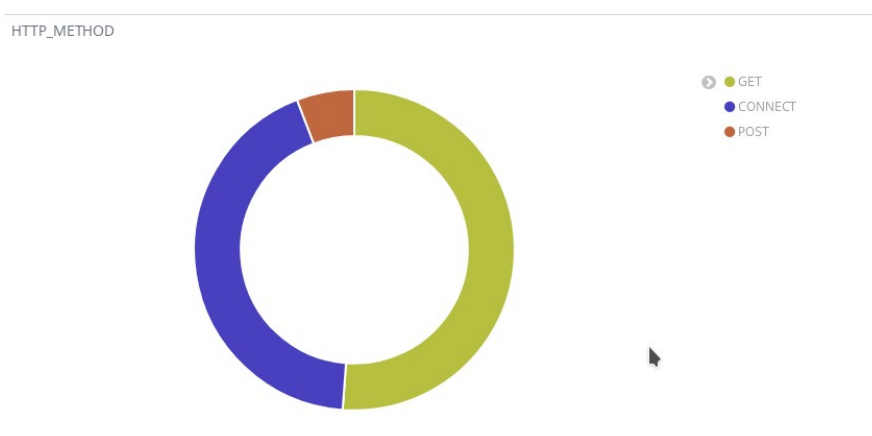
*Figura 41: Opción Visualize en Kibana*

En dicho menú es posible incluir nuevas visualizaciones si accedemos al entorno de creación por el icono “+”. Una vez dentro nos solicitará que tipo de visualización deseamos crear, entre los distintos tipos se dispone de tipo área, mapa, tabla de datos, tarta, barras horizontales o verticales entre otros. Dependiendo del tipo seleccionado muestra distintas opciones de filtrado.

Para la realización de este trabajo se han creado 6 visualizaciones distintas que se mostrarán con un único dashboard. Las visualizaciones son las siguientes:

- HTTP\_METHOD – Tipo Tarta

Muestra un diagrama de tarta según los métodos HTTP solicitados a los proxys.



*Figura 42: Visualización HTTP\_METHOD*

- HTTP\_STATUS\_CODE – Tipo Tarta

Muestra un diagrama de tarta según los códigos de respuesta HTTP ofrecidos por los servidores proxy.

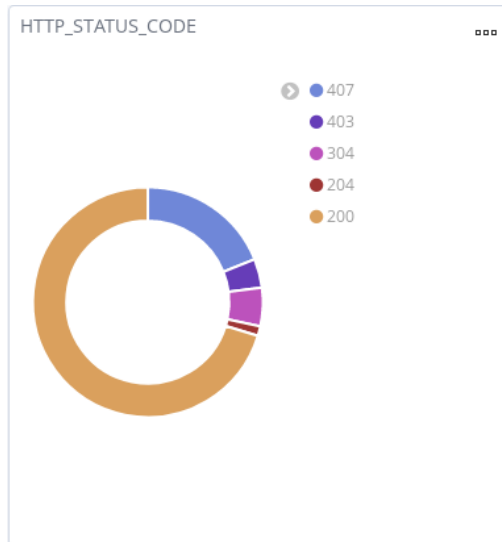


Figura 43: Visualización HTTP\_STATUS\_CODE

- SQUID\_REQUEST\_STATUS – Tipo Tarta

Muestra un diagrama de tarta según la respuesta ofrecida por el servidor proxy de acceso a su cache.

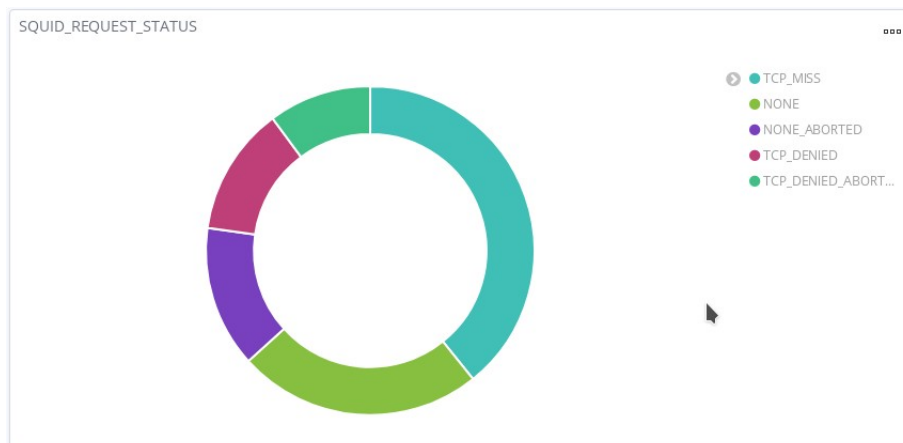


Figura 44: Visualización SQUID\_REQUEST\_STATUS

- Peticiones por Proxy – Tipo Barras Verticales

Muestra un diagrama de barras donde se muestran la cantidad de peticiones servidas por cada nodo proxy.

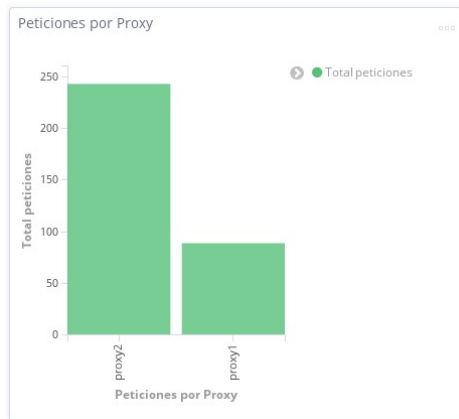


Figura 45: Visualización Peticiónes por Proxy

- Top 10 Web Solicitadas – Tipo Tarta

Muestra un diagrama de tarta donde se muestran las 10 páginas web más solicitadas a los servidores proxy.

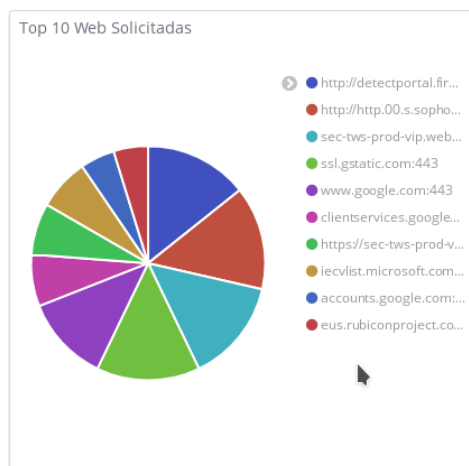


Figura 46: Visualización Top 10 Web Solicitadas

- Top 10 Usuarios – Tipo Tarta

Muestra un diagrama de tarta donde se muestran los 10 usuarios que solicitan más peticiones. En este caso la imagen sólo muestra el usuario utilizado para las pruebas.

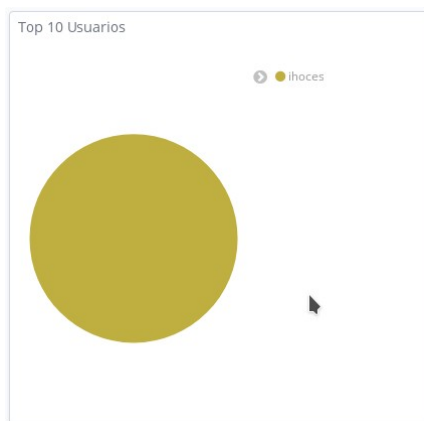


Figura 47: Visualización Top 10 Usuarios

Una vez se han creado las distintas visualizaciones podemos proceder a integrarlas en un único panel o **dashboard**. Para ello accedemos al menú anterior, en la opción situada debajo de la antes utilizada.

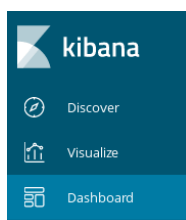


Figura 48: Opción Dashboard en Kibana

En este menú podemos proceder a crear un nuevo panel con el botón "Create new dashboard". En la nueva pantalla podremos añadir las visualizaciones creadas antes, con el botón "add".

El panel creado para este proyecto se ha denominado "Panel de Control" y tiene el aspecto mostrado en la siguiente imagen.

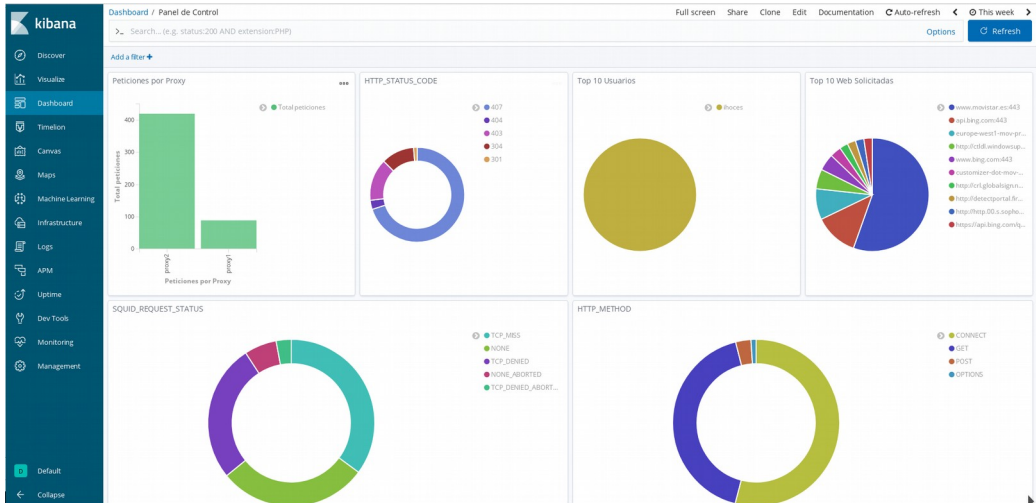


Figura 49: Dashboard en Kibana

Todos los elementos creados en este apartado, tanto visualizaciones como el panel o dashboard pueden ser exportados en formato fichero json y se han incluido en el ANEXO 7.

## 5. Configuración de acceso de los usuarios.

Una vez el sistema se encuentra instalado y en funcionamiento sólo resta configurar el **acceso en los equipos clientes**.

Dado que se ha optado por una configuración de Proxy en modo explícito, es necesario facilitar a los clientes la dirección y el puerto por el que pueden conectarse.

La configuración en la parte cliente reside en indicar en los navegadores que puedan utilizar los usuarios la dirección del sistema balanceada, "inet.agenciaagrariaypesquera.es" y el puerto TCP 3128.

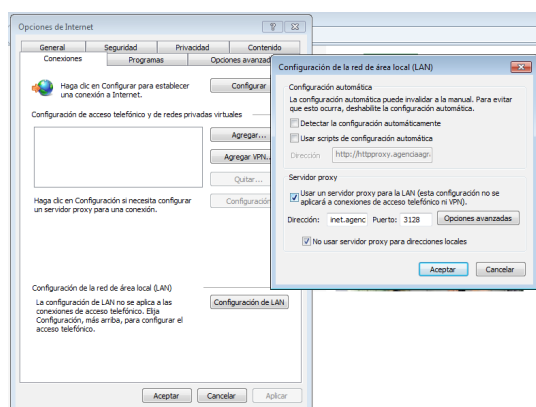


Figura 50: Configuración en Internet Explorer

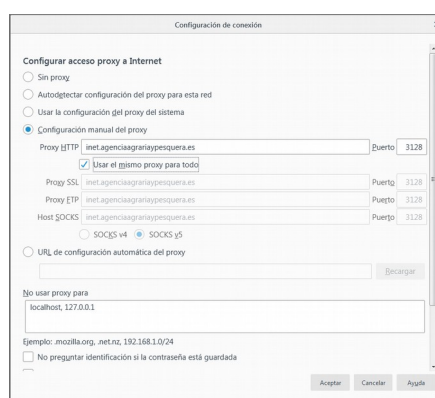


Figura 51: Configuración en Mozilla Firefox

Con esta configuración los navegadores ya pueden realizar las peticiones web al sistema proxy-cache.

Esta configuración es básica para el funcionamiento del sistema, pero es posible mejorarla con la incorporación de un sistema de **fichero PAC**. Este sistema se basa en la utilización de un fichero localizado en una URL, el cual, al ser descargado es ejecutado por el navegador para determinar su configuración proxy.

Mediante la utilización de este fichero, se pueden establecer comportamientos distintos según los clientes o destinos que hagan uso de él. Por ejemplo, se puede establecer que en caso de tratarse de una petición a una URL localizada en el direccionamiento local de la red, esta no sea trasladada a los proxys y que se realice directamente.

Una configuración adicional al sistema es implementar el **sistema WPAD**(Web Proxy Auto-Discovery Protocol) [41], el cual se utiliza para que el fichero PAC indicado antes, sea localizado de forma automática en la red. Esta configuración se puede realizar gracias a DHCP o mediante DNS.

En el caso de DHCP, el servicio debe almacenar la localización del fichero PAC para que cuando facilite las direcciones IP se les informe también de la misma.

En cambio, DNS WPAD, hace uso de un método de descubrimiento en la red. En este caso, el sistema busca una URL encabezada por "wpad." dentro del dominio y con el fichero wpad.dat que contiene el PAC.

El formato de un fichero PAC (Proxy auto-config) contiene una **función en JavaScript** llamada "FindProxyForURL". Esta función devuelve una cadena con el método de acceso que debe utilizar el navegador.

Una fichero PAC simple tendría el siguiente contenido:

```
function FindProxyForURL(url, host)
{
    return "PROXY inet.agenciaagrariaypesquera.es:3128; DIRECT";
}
```

Si queremos establecer una configuración más elaborada se puede recurrir a distintas funciones de análisis de la dirección IP de destino u origen, o de la URL a la que se desea acceder.

El fichero WPAD.DAT utilizado para la instalación realizada es el siguiente:

```
function FindProxyForURL(url, host)
{
    if ( (isInNet (dnsResolve(host), "10.0.0.0", "255.0.0.0")) ||
        (isInNet (dnsResolve(host), "192.168.0.0", "255.255.0.0")) ||
        (isInNet (dnsResolve(host), "127.0.0.1", "255.255.255.255")))
        return "DIRECT";
    else
        return "PROXY inet.agenciaagrariaypesquera.es:3128";
}
```

En este fichero se analiza la dirección IP a la que se desea acceder. Mediante la función dnsResolve se obtiene la dirección IP de destino solicitada. Esta información es pasada a la función isInNet la cual verifica si esta dirección IP forma parte de la subred que se le pasa como dos parámetros, dirección de red y máscara de subred. Si la dirección se encuentra en alguno de estos direccionamientos locales, la petición es enviada directamente por el navegador. En caso contrario, se hace uso del proxy.



## 6. Implantación y servicios en Cloud.

### 6.1. Funcionamiento de los productos elegidos en modo Cloud.

Todos los productos que se han utilizado para la implementación de este sistema, se pueden instalar en sistemas Linux. Dependiendo de la distribución escogida, los paquetes de software necesarios pueden variar.

Dado esto, siempre se puede contratar un servicio de cloud en el que se puedan desplegar cada uno de los componentes. Dependiendo de la modalidad de cloud que se escoja, variará la dificultad de la implantación.

En caso de elegir una modalidad de **Infraestructura como servicio** (IaaS), no solo será necesario ejecutar los pasos indicados anteriormente sino que conllevará la gestión de toda la infraestructura a un nivel mas bajo.

Por otro lado, si se utiliza la modalidad de **Plataforma como servicio** (PaaS), sólo requeriría la instalación de los servidores virtuales mas la configuración indicada en los apartados anteriores.

Por último, la modalidad de **Software como servicio** (SaaS), es la modalidad que quizá fuese de mayor complejidad configurar. En este caso, sería necesario localizar un servicio de cloud que ofreciese el servicio **Squid** pero con las funcionalidades que se han especificado. El servicio de balanceo de carga, al no presentar mucha complejidad a nivel de requisitos, puede ser implementado con otro servicio de balanceo que no fuese **Haproxy**, si este no estuviera disponible. Por otro lado, el **Stack Elk** puede contratarse directamente con el propio fabricante en su Cloud.

En todos estos casos sería necesario una correcta configuración de visibilidad y acceso entre todos los componentes que forman parte del sistema y con los usuarios que los utilizan.

Por otro lado, hay que tener en cuenta que muchos servicios de cloud facturan por **ancho de banda consumido**. Un sistema de navegación web con almacenamiento de logs puede conllevar un consumo muy alto de este recurso. En consecuencia, es necesario realizar un estudio pormenorizado del consumo estimado y de su coste a fin de evaluar la viabilidad de dicha implantación.

### 6.2. Servicio DNS inteligente.

En el mercado existe otro servicio que puede dotar al sistema de una capa mayor de seguridad. Este servicio es el de filtrado y control de acceso mediante el servicio de DNS. Gracias a ello, se pueden realizar

bloqueos sobre dominios o direcciones IP al realizar la consulta DNS. Estos servicios suelen estar **ofrecidos en modalidad cloud** y aplican técnicas mas avanzadas de análisis de comportamiento mas allá del bloqueo por listas negras.

Estos servicios conllevan un **coste por usuario** que realiza la conexión. A continuación se describen a modo de muestra, un par de alternativas con sus costes.

## CISCO UMBRELLA

Este servicio pertenece a la compañía Cisco [42]. Este, provee acceso seguro a usuarios, tanto si se encuentran en la red de la compañía o si utilizan una red ajena.

Hace uso de la tecnología de OpenDNS con motores de seguridad y aprendizaje automático con el objetivo de bloquear amenazas. Este bloqueo se realiza en las **capas DNS e IP**.

De forma adicional a esta funcionalidad, Cisco incorpora motores de protección frente a virus y malware.

Una funcionalidad a destacar, es la aplicación de **modelos predictivos sobre dominios**. Con ello, es capaz de predecir las mutaciones de dominios DNS que pueden ser utilizados en futuras campañas catalogándolos como maliciosos.

Para asegurar las conexiones a internet desde redes no seguras con equipos portátiles de la compañía, el sistema implementa un cliente ligero que permitirá el acceso al servicio y recopilará la información de accesos realizados.

Este producto viene derivado del servicio **OpenDNS**[43] que fue adquirido por Cisco. En este caso, se establece una diferenciación según el cliente objetivo. OpenDNS se centra mas en el **uso personal y de pequeña empresa** y Cisco Umbrella[44] está orientado a compañías de mayor tamaño con mayores exigencias de rendimiento.

En el momento de la realización de este TFG, el **coste por usuario** de acceso al servicio de OpenDNS en la modalidad SMALL BUSINESS corresponde a 20\$ al año para entre 1 usuarios y 3 dispositivos. Por otro lado, el coste del servicio con Cisco Umbrella que para 100 usuarios al mes corresponde a 2.70\$.

## TitanHQ

Esta compañía ofrece otra alternativa para el filtrado de contenido web mediante el uso de control del servicio DNS en cloud[45].

Entre las características que ofrece el producto se encuentran las siguientes:

- Bloqueo de Malware.
- Filtrado de contenidos.
- Gestión de políticas de manera flexible.
- Securización de los dispositivos ajenos a la compañía(BYOD).
- Generación de informes.
- Escalabilidad

En este caso, TitanHQ ofrece la **modalidad de instalación local** de la solución en la infraestructura del cliente en modalidad appliance. Esta modalidad se denomina WebTitan Gateway. Este contiene una base de datos de direcciones URL categorizadas para el filtrado.

Otra de las modalidades a destacar es la denominada WebTitan Cloud. En este caso, el sistema se encuentra desplegado en la cloud del fabricante del producto.

En cuanto al **coste de acceso** para 100 usuarios durante un mes es de 1.10\$. El mismo fabricante ofrece varias comparativas con el producto del fabricante Cisco[46].

En ambos casos, la integración con el sistema previamente instalado se realiza mediante la configuración en cada nodo proxy de las direcciones IP de Umbrella o TitanHQ como DNS de los nodos. De esta forma, las resoluciones de las peticiones realizadas por los usuarios a los nodos proxy serán tratadas por el servicio contratado.

El modo de instalación de producto mas adecuado al sistema planteado es el de Virtual Appliance. De esta forma, se puede indicar al sistema que en caso de realizarse peticiones internas, estas sean derivadas hacia los servidores DNS internos.

Estos Appliance son los encargados de conectar con los servicios en cloud del proveedor del producto.

## 7. Conclusiones

Tras la realización de este trabajo, he detectado las siguientes consideraciones:

- Al ser un proyecto no enfocado a un entorno concreto, sino de propósito general, que abarcase tanto a pequeños como a grandes entornos, ha sido necesario analizar distintas alternativas que permitiesen una fácil escalabilidad.
- Al centrarse el proyecto en productos de origen open source, se han detectado muchos proyectos que nacieron con grandes expectativas pero que al final se quedaron en una fase inicial o embrionaria.
- Existen muchas alternativas a la solución planteada en este TFG. Las que requieren un coste de inversión, el cual, puede llegar a variar mucho dependiendo de la fiabilidad de la marca del mismo o del volumen de datos o usuarios que vaya a soportar.
- Los proyectos con gran nivel de madurez e implantación como Squid-Cache, gozan de una amplio abanico de recursos de soporte fáciles de consultar. De esta forma se mitiga el echo de no contar con un soporte oficial del producto. A la hora de optar por una solución de este tipo es importante analizar estos recursos y el tamaño y experiencia de la comunidad que lo soporta.
- Gracias a la estructura con la que se ha creado el proyecto Squid-Cache, sería fácil implementar complementos específicos para entornos concretos, ya sean módulos de autenticación o plugin de análisis y transformación de URL.
- A nivel de planificación del proyecto, ha sido posible seguir una estructura lógica de progresión del mismo. El trabajo pudo dividirse en tareas lógicas y bien estructuradas, las cuales fueron programadas con un margen suficiente de tiempo de dedicación. En las situaciones en las que se han detectado problemas de funcionamiento del sistema ha sido posible buscar soluciones a las mismas de forma rápida para ser probadas con garantías.
- Aun cuando el sistema se ha montado y es totalmente operativo, restaría analizar la fortaleza del sistema ante cargas altas de trabajo. Sería necesario buscar y analizar distintas herramientas de pruebas de carga para tras escoger una, realizar un plan de pruebas específico cuyos resultados pudieran ofrecer una estimación realista del comportamiento global del sistema.
- Uno de los componentes que requerirían un análisis mayor, es la estructura de cache de los nodos proxy. Por defecto, cada nodo mantiene su propia cache de navegación, pero es posible crear una estructura jerarquizada y compartida para minimizar las búsquedas fuera de esta memoria.

## 8. Glosario

- **AGAPA** – Agencia de Gestión Agraria y Pesquera de Andalucía.
- **PROXY** – Servicio de concentración de tráfico de navegación web.
- **CACHE** – Sistema de almacenamiento de acceso común.
- **KERBEROS** – Protocolo de autenticación utilizado por Microsoft.
- **LDAP** – Protocolo de acceso ligero a directorios.
- **HTTP** – Protocolo de transferencia de hipertexto.
- **HTTPS** – Protocolo seguro de transferencia de hipertexto.
- **SQUID** – Proxy – Cache que soporta HTTP, HTTPS, FTP entre otros.
- **TIC** – Tecnologías de la Información y la Comunicación.
- **MITM** – Man in the middle
- **LOG** – Historial o registro
- **SSL** – Secure Socket Layer, Protocolo de encriptación y desencriptación.
- **TFG** – Trabajo Fin de Grado.
- **HAPROXY** – Balanceador de carga y Proxy
- **ELASTICSEARCH** – Servidor de búsqueda basado en Lucene.
- **KIBANA** – Herramienta de visualización de datos.
- **LOGSTASH** – Herramienta para recolectar y parsear logs.
- **BEATS / FILEBEAT** – Herramienta para recopilar y transmitir datos.
- **SOCKS** – Protocolo de Internet transparente a Firewall.
- **FTP** – File Transfer Protocol. Protocolo de transferencia de ficheros.
- **FIREWALL** – Cortafuegos, dispositivo que controla la comunicación entre dispositivos o redes.
- **FRONTEND** – Parte de un sistema encargado de interactuar con los usuarios y cuya información transmite a los Backend.
- **BACKEND** – Parte de un sistema encargado de realizar las operaciones solicitadas cuyos resultados son devueltos a un Frontend.
- **OPEN SOURCE** – Código abierto, modelo de desarrollo de software colaborativo.
- **ACTIVE DIRECTORY** – Servicio de Directorio de Microsoft.
- **ACL** – Lista de control de acceso.
- **NTP** – Protocolo de tiempo de red.
- **DNS** – Sistema de nombres de dominio.
- **KEYTAB** – Tabla de llaves, en formato clave:valor.
- **NTLM** – Protocolos de seguridad de Microsoft.
- **CCN-CERT** - Centro Criptológico Nacional Computer Emergency Response Team
- **OPENSSL** – Conjunto de herramientas con funciones criptográficas.
- **VMWARE** – Fabricante de software de virtualización de servidores.
- **VSPHERE** – Software de virtualización fabricado por Vmware.
- **VRRP** – Protocolo de enrutamiento virtual redundante.
- **WPAD** – Protocolo de auto-descubrimiento proxy web.
- **PAC** – Auto configuración Proxy.
- **BYOD** – Bring your own device(Traer tu propio dispositivo).
- **IAAS** – Infraestructura como servicio.
- **PAAS** – Plataforma como servicio.
- **SAAS** – Software como servicio.

## 9. Bibliografía

- 1: Symantec Corporation - ProxySg and Advanced Secure Gateway, <https://www.symantec.com/es/es/products/proxy-sg-and-advanced-secure-gateway>, 10/03/2019
- 2: Apache Traffic Server Documentation, <https://docs.trafficserver.apache.org/en/latest/index.html>, 10/03/2019
- 3: Haproxy, <http://www.haproxy.org>, 10/03/2019
- 4: Privoxy, <http://www.privoxy.org/>, 11/02/19
- 5: Nginx, <https://www.nginx.com/>, 11/03/2019
- 6: Varnish, <https://varnish-cache.org/>, 11/03/2019
- 7: Duane Wessels, Squid: The Definitive Guide, O'Reilly2009
- 8: Kulbir Saini, Squid Proxy Server 3.1: Beginner's Guide, Packt>2011
- 9: Wiki Squid-Cache, <https://wiki.squid-cache.org/>, 12/03/2019
- 10: Seesaw, <https://opensource.google.com/projects/seesaw>, 13/03/2019
- 11: LoadMaster by Kemp, <https://kemptechnologies.com/es/virtual-load-balancer/>, 02/04/2019
- 12: Zevenet, <https://www.zevenet.com/es/productos/empresa/virtual/>, 02/03/2019
- 13: Neutrino, <http://neutrinoslb.github.io/>, 02/03/2019
- 14: Balance by InLab, <https://www.inlab.net/balance/>, 02/03/2019
- 15: Traefik, <https://traefik.io/>, 02/03/2019
- 16: F5, <https://f5.com/es/products/big-ip>, 02/03/2019
- 17: Barracuda, <https://www.barracuda.com/products/loadbalancer>, 02/03/2019
- 18: TP-Link, <https://www.tp-link.com/es/business-networking/load-balance-router/>, 02/03/2019
- 19: Fortinet, <https://www.fortinet.com/products/application-delivery-controller/fortiadc.html>, 02/03/2019
- 20: Clamav, <http://www.clamav.net>, 15/02/2019
- 21: Squidclamav, <http://squidclamav.darold.net/>, 15/02/2019
- 22: Viralator, <http://viralator.sourceforge.net/>, 16/02/2019
- 23: Elastic Stack, <https://www.elastic.co/es/products/>, 12/02/2019
- 24: Elasticsearch, <https://www.elastic.co/es/products/elasticsearch>, 12/02/2019
- 25: Lucene, <http://lucene.apache.org/>, 12/02/2019
- 26: Logstash, <https://www.elastic.co/es/products/logstash>, 12/02/2019
- 27: Kibana, <https://www.elastic.co/es/products/kibana>, 12/02/2019

- 28: Squid-Cache V4, <http://www.squid-cache.org/Versions/v4/>, 15/02/2019
- 29: Squid Kerberos Authentication Helper, <https://sourceforge.net/projects/squidkerbauth/>, 15/02/2019
- 30: Centro Criptológico Nacional, <https://www.ccn-cert.cni.es/>, 10/03/2019
- 31: CCN-STIC 660 Seguridad en Proxy, <https://www.ccn-cert.cni.es/comunicacion-eventos/comunicados-ccn-cert/5946-configuracion-segura-de-servidores-proxy.html>, 10/03/2019
- 32: Generador de certificados, [http://www.squid-cache.org/Doc/config/sslcrt\\_d\\_program/](http://www.squid-cache.org/Doc/config/sslcrt_d_program/), 12/03/2019
- 33: Internet Content Adaptation Protocol, RFC 3507, <http://www.rfc-editor.org/rfc/rfc3507.txt>, 14/03/2019
- 34: Virtual Router Redundancy Protocol, <https://tools.ietf.org/html/rfc5798>, 16/03/2019
- 35: RC4-HMAC Encriptado Kerberos, <https://tools.ietf.org/html/rfc4757>, 18/03/2019
- 36: Elastic, <https://www.elastic.co/>, 25/03/2019
- 37: Beats, <https://www.elastic.co/products/beats>, 26/03/2019
- 38: Página de Descargas Elastic, <https://www.elastic.co/downloads/>, 26/03/2019
- 39: Estructura del fichero de configuración de Logstash, <https://www.elastic.co/guide/en/logstash/current/configuration-file-structure.html>, 26/03/2019
- 40: Parámetro logformat Squid, <http://www.squid-cache.org/Doc/config/logformat/>, 27/03/2019
- 41: Web Proxy Auto-Discovery Protocol, <https://tools.ietf.org/html/draft-ietf-wrec-wpad-01>, 08/04/2019
- 42: Cisco, [www.cisco.com](http://www.cisco.com), 23/04/2019
- 43: OpenDNS, [www.opendns.com](http://www.opendns.com), 23/04/2019
- 44: Cisco Umbrella, <https://umbrella.cisco.com/>, 23/04/2019
- 45: WebTitan, <https://www.titanhq.com/webtitan>, 23/04/2019
- 46: webtitan-cloud-verus-opendns, <https://www.titanhq.com/webtitan-cloud-verus-opendns>, 23/04/2019

# 10. Anexos

## ANEXO 1: Script arranque servicio Squid

```
#!/bin/sh
#
# squid4          Startup script for the SQUID HTTP proxy-cache.
#
# Version:        squid4
#
### BEGIN INIT INFO
# Provides:       squid4
# Required-Start: $network $remote_fs $syslog
# Required-Stop:  $network $remote_fs $syslog
# Should-Start:   $named
# Should-Stop:    $named
# Default-Start:  2 3 4 5
# Default-Stop:   0 1 6
# Short-Description: Squid HTTP Proxy version 4.x
### END INIT INFO

##### MODIFICACION PARA KEYTAB DE KERBEROS #####

KRB5_KTNAME=/opt/squid46/etc/PROXY.keytab
KRB5RCACHETYPE=none
export KRB5_KTNAME
export KRB5RCACHETYPE

NAME=squid46
DESC="Squid HTTP Proxy 4.x"
DAEMON=/opt/squid46/sbin/squid
PIDFILE=/var/run/$NAME.pid
CONFIG=/opt/squid46/etc/squid.conf
SQUID_ARGS="-YC -f $CONFIG"

[ ! -f /etc/default/squid4 ] || ./etc/default/squid4

./lib/lsb/init-functions

PATH=/bin:/usr/bin:/sbin:/usr/sbin

[ -x $DAEMON ] || exit 0

ulimit -n 65535

find_cache_dir () {
    w=" " # space tab
    res=`sed -ne '
        s/^[^$w]*\+([^\$w]*)\+([^\$w]*)\+([^\$w]*)\+).*$\1/p;
        t end;
        d;
        :end q' < $CONFIG`
    [ -n "$res" ] || res=$2
    echo "$res"
}

find_cache_type () {
    w=" " # space tab
    res=`sed -ne '
        s/^[^$w]*\+([^\$w]*)\+).*$\1/p;
        t end;
        d;
        :end q' < $CONFIG`
    [ -n "$res" ] || res=$2
    echo "$res"
}

start () {
    cache_dir=`find_cache_dir cache_dir /var/spool/squid4`
    cache_type=`find_cache_type cache_dir ufs`
```



```

#
# Create spool dirs if they don't exist.
#
if [ "$cache_type" = "coss" -a -d "$cache_dir" -a ! -f "$cache_dir/stripes" ] || [ "$cache_type" != "coss" -a -d
"$cache_dir" -a ! -d "$cache_dir/00" ]
then
    log_warning_msg "Creating $DESC cache structure"
    $DAEMON -z
fi

umask 027
ulimit -n 65535
cd $cache_dir
start-stop-daemon --quiet --start \
    --pidfile $PIDFILE \
    --exec $DAEMON -- $SQUID_ARGS < /dev/null
return $?
}

stop () {
    PID=`cat $PIDFILE 2>/dev/null`
    start-stop-daemon --stop --quiet --pidfile $PIDFILE --exec $DAEMON
    #
    # Now we have to wait until squid has _really_ stopped.
    #
    sleep 2
    if test -n "$PID" && kill -0 $PID 2>/dev/null
    then
        log_action_begin_msg " Waiting"
        cnt=0
        while kill -0 $PID 2>/dev/null
        do
            cnt=`expr $cnt + 1`
            if [ $cnt -gt 24 ]
            then
                log_action_end_msg 1
                return 1
            fi
            sleep 5
            log_action_cont_msg ""
        done
        log_action_end_msg 0
        return 0
    else
        return 0
    fi
}

case "$1" in
start)
    log_daemon_msg "Starting $DESC" "$NAME"
    if start ; then
        log_end_msg $?
    else
        log_end_msg $?
    fi
    ;;
stop)
    log_daemon_msg "Stopping $DESC" "$NAME"
    if stop ; then
        log_end_msg $?
    else
        log_end_msg $?
    fi
    ;;
reload|force-reload)
    log_action_msg "Reloading $DESC configuration files"
    start-stop-daemon --stop --signal 1 \
        --pidfile $PIDFILE --quiet --exec $DAEMON
    log_action_end_msg 0
    ;;
restart)
    log_daemon_msg "Restarting $DESC" "$NAME"
    stop
    if start ; then
        log_end_msg $?
    fi
}

```

```

        else
            log_end_msg $?
        fi
    ;;
*)
    echo "Usage: /etc/init.d/$NAME {start|stop|reload|force-reload|restart}"
    exit 3
    ;;
esac

exit 0

```

## ANEXO 2: Respuesta unión con Kerberos, comando msktutil

```

-- init_password: Wiping the computer password structure
-- generate_new_password: Generating a new, random password for the computer account
-- generate_new_password: Characters read from /dev/urandom = 89
-- create_fake_krb5_conf: Created a fake krb5.conf file: /tmp/.msktrb5.conf-P9bRHc
-- reload: Reloading Kerberos Context
-- finalize_exec: SAM Account Name is: PROXY1-K$
-- try_machine_keytab_princ: Trying to authenticate for PROXY1-K$ from local keytab...
-- try_machine_keytab_princ: Error: krb5_get_init_creds_keytab failed (Client not found in Kerberos database)
-- try_machine_keytab_princ: Authentication with keytab failed
-- try_machine_keytab_princ: Trying to authenticate for PROXY1-K$ from local keytab...
-- try_machine_keytab_princ: Error: krb5_get_init_creds_keytab failed (Client not found in Kerberos database)
-- try_machine_keytab_princ: Authentication with keytab failed
-- try_machine_keytab_princ: Trying to authenticate for host/proxy1.dap.es from local keytab...
-- try_machine_keytab_princ: Error: krb5_get_init_creds_keytab failed (Client not found in Kerberos database)
-- try_machine_keytab_princ: Authentication with keytab failed
-- try_machine_password: Trying to authenticate for PROXY1-K$ with password.
-- create_default_machine_password: Default machine password for PROXY1-K$ is proxy1-k
-- try_machine_password: Error: krb5_get_init_creds_keytab failed (Client not found in Kerberos database)
-- try_machine_password: Authentication with password failed
-- try_user_creds: Checking if default ticket cache has tickets...
-- finalize_exec: Authenticated using method 5
-- LDAPConnection: Connecting to LDAP server: dc1.dap.es
SASL/GSSAPI authentication started
SASL username: administrador@DAP.ES
SASL SSF: 56
SASL data security layer installed.
-- ldap_get_base_dn: Determining default LDAP base: dc=DAP,dc=ES
-- ldap_check_account: Checking that a computer account for PROXY1-K$ exists
-- ldap_create_account: Computer account not found, create the account
No computer account for PROXY1-K found, creating a new one.
-- ldap_check_account_strings: Inspecting (and updating) computer account attributes
-- ldap_check_account_strings: Found userPrincipalName =
-- ldap_check_account_strings: userPrincipalName should be HTTP/proxy1.dap.es@DAP.ES
-- ldap_set_userAccountControl_flag: Setting userAccountControl bit at 0x200000 to 0x0
-- ldap_set_userAccountControl_flag: userAccountControl not changed 0x1000
-- ldap_get_kvno: KVNO is 1
-- ldap_add_principal: Checking that adding principal HTTP/proxy1.dap.es to PROXY1-K$ won't cause a conflict
-- ldap_add_principal: Adding principal HTTP/proxy1.dap.es to LDAP entry
-- ldap_add_principal: Checking that adding principal host/proxy1.dap.es to PROXY1-K$ won't cause a conflict
-- ldap_add_principal: Adding principal host/proxy1.dap.es to LDAP entry
-- execute: Updating all entries for proxy1.dap.es in the keytab WRFILE:/opt/squid46/etc/PROXY.keytab
-- update_keytab: Updating all entries for PROXY1-K$
-- add_principal_keytab: Adding principal to keytab: PROXY1-K$
-- add_principal_keytab: Using salt of DAP.EShostproxy1-k.dap.es
-- add_principal_keytab: Adding entry of enctype 0x17
-- add_principal_keytab: Using salt of DAP.EShostproxy1-k.dap.es
-- add_principal_keytab: Adding entry of enctype 0x11
-- add_principal_keytab: Using salt of DAP.EShostproxy1-k.dap.es
-- add_principal_keytab: Adding entry of enctype 0x12
-- add_principal_keytab: Adding principal to keytab: PROXY1-K$
-- add_principal_keytab: Removing entries with kvno < 0
-- add_principal_keytab: Deleting PROXY1-K$@DAP.ES kvno=1, enctype=23
-- add_principal_keytab: Deleting PROXY1-K$@DAP.ES kvno=1, enctype=17
-- add_principal_keytab: Deleting PROXY1-K$@DAP.ES kvno=1, enctype=18
-- add_principal_keytab: Using salt of DAP.EShostproxy1-k.dap.es
-- add_principal_keytab: Adding entry of enctype 0x17
-- add_principal_keytab: Using salt of DAP.EShostproxy1-k.dap.es
-- add_principal_keytab: Adding entry of enctype 0x11
-- add_principal_keytab: Using salt of DAP.EShostproxy1-k.dap.es
-- add_principal_keytab: Adding entry of enctype 0x12

```

```

-- add_principal_keytab: Adding principal to keytab: HTTP/proxy1.dap.es
-- add_principal_keytab: Removing entries with kvno < 0
-- add_principal_keytab: Using salt of DAP.EShostproxy1-k.dap.es
-- add_principal_keytab: Adding entry of enctype 0x17
-- add_principal_keytab: Using salt of DAP.EShostproxy1-k.dap.es
-- add_principal_keytab: Adding entry of enctype 0x11
-- add_principal_keytab: Using salt of DAP.EShostproxy1-k.dap.es
-- add_principal_keytab: Adding entry of enctype 0x12
-- add_principal_keytab: Adding principal to keytab: host/PROXY1-K
-- add_principal_keytab: Removing entries with kvno < 0
-- add_principal_keytab: Using salt of DAP.EShostproxy1-k.dap.es
-- add_principal_keytab: Adding entry of enctype 0x17
-- add_principal_keytab: Using salt of DAP.EShostproxy1-k.dap.es
-- add_principal_keytab: Adding entry of enctype 0x11
-- add_principal_keytab: Using salt of DAP.EShostproxy1-k.dap.es
-- add_principal_keytab: Adding entry of enctype 0x12
-- update_keytab: Entries for SPN HTTP/proxy1.dap.es have already been added. Skipping ...
-- add_principal_keytab: Adding principal to keytab: host/proxy1.dap.es
-- add_principal_keytab: Removing entries with kvno < 0
-- add_principal_keytab: Using salt of DAP.EShostproxy1-k.dap.es
-- add_principal_keytab: Adding entry of enctype 0x17
-- add_principal_keytab: Using salt of DAP.EShostproxy1-k.dap.es
-- add_principal_keytab: Adding entry of enctype 0x11
-- add_principal_keytab: Using salt of DAP.EShostproxy1-k.dap.es
-- add_principal_keytab: Adding entry of enctype 0x12
-- ~KRB5Context: Destroying Kerberos Context

```

## ANEXO 3: Script actumalware

```

#!/bin/bash
RUTASQUID="/opt/squid46/etc/acls/"

# Obtenemos el listado de listas negras
cd $RUTASQUID
/usr/bin/wget -N http://mirror1.malwaredomains.com/files/domains.txt --no-if-modified-since
/usr/bin/wget -N https://www.spamhaus.org/drop/edrop.txt --no-if-modified-since
/usr/bin/wget -N https://www.spamhaus.org/drop/drop.txt --no-if-modified-since
/usr/bin/wget -N https://www.squidblacklist.org/downloads/squid-malicious.acl --no-if-modified-since

# Borramos el fichero de Malware.txt ya que será regenerado al final de este script.

rm -f $RUTASQUID/DOM_Malware.txt
rm -f $RUTASQUID/IP_Malware.txt

#Tratamos la lista de Dominios Malware
# Quitamos los comentarios
sed '/#/d' $RUTASQUID/domains.txt > $RUTASQUID/domains2.txt

# Nos quedamos con la 1ª y 2ª columna que son los datos que nos interesan
cat $RUTASQUID/domains2.txt | awk '{print $1"\t"$2}' > $RUTASQUID/domains3.txt

# Leemos el fichero línea a línea y separamos las columnas en diferentes líneas para poder filtrar
for line in $(cat $RUTASQUID/domains3.txt); do echo "$line" >> $RUTASQUID/domains4.txt ; done

# Filtramos las líneas que contienen "."
sed -n -e '/\./p' $RUTASQUID/domains4.txt > $RUTASQUID/domains5.txt

# Insertamos "." a todas las líneas para formato de squid
while read linea
do
    echo ".$linea >> $RUTASQUID/Malware.txt
done < $RUTASQUID/domains5.txt

sed '1,5 d' squid-malicious.acl | awk '{print $1}' > Malicius.txt

cat $RUTASQUID/Malware.txt $RUTASQUID/Malicius.txt > DOM_Malware.txt

#Tratamos las IPs Malware
sed '1,4 d' $RUTASQUID/drop.txt | awk '{print $1}' > $RUTASQUID/IP_drop.txt
sed '1,4 d' $RUTASQUID/edrop.txt | awk '{print $1}' > $RUTASQUID/IP_edrop.txt

```

```
cat $RUTASQUID/IP_drop.txt $RUTASQUID/IP_edrop.txt > $RUTASQUID/IP_Malware.txt
```

```
# Borramos ficheros intermedios de este script.
```

```
rm -f $RUTASQUID/domains2.txt $RUTASQUID/domains3.txt $RUTASQUID/domains4.txt $RUTASQUID/domains5.txt  
$RUTASQUID/domains.txt $RUTASQUID/IP_drop.txt $RUTASQUID/IP_edrop.txt $RUTASQUID/Malicious.txt
```

```
# Cargamos las acls en squid
```

```
/opt/squid46/sbin/squid -k reconfigure
```

## ANEXO 4: Código fuente de una página de error personalizada

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">  
<html>  
  <head>  
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">  
    <title>ERROR: Acceso Denegado</title>  
    <link rel="stylesheet" type="text/css" href="http://proxy1.dap.es:81/css/estilosError.css" />  
  </head>  
  <body>  
    <div id="banner">  
      <a href="http://www.agenciaagrariaypesquera.es"></a>  
    </div>  
    <div id="content">  
      <div id="error">Lo sentimos, la página <a href="%U"%U</a> no le ha podido ser entregada por el  
siguiente motivo:  
      </div>  
        
      <div id="ticket">  
        Si considera que ha sido redireccionado a esta página por error o que la página solicitada debiera  
ser visible, puede notificarlo pulsando <a href="https://www.agapa.junta-andalucia.es/autoservicio/">Aqui</a>,  
seleccionando el SERVICIO: INTERNET y la OPERACION: Solicitar Acceso.  
      </div>  
    </div>  
  </body>  
</html>
```

## ANEXO 5: keepalived.conf

```
# config file for keepalived on LB01
```

```
global_defs {  
  notification_email {  
    cuenta@dominio.es  
  }  
  
  notification_email_from cuenta@dominio.es  
  smtp_server 10.239.212.109  
  smtp_connect_timeout 30  
  router_id lb01  
}  
  
vrrp_script chk_haproxy {  
  script "killall -0 haproxy"  
  timeout 5          #Tiempo máximo de respuesta  
  interval 2        #Chequeo cada X segundos  
  fail 4             #Numero de intentos para marcar fallido  
  rise 2             #Numero aciertos para retomar el servicio  
  weight 2  
}  
  
vrrp_instance VI_1 {  
  state MASTER  
  interface ens192  
  smtp_alert  
  virtual_router_id 99  
  priority 110      # master: 110, slave: 109  
  advert_int 1
```

```

authentication {
    auth_type PASS
    auth_pass XXXXXXXX # use 8 chars & something better
}

virtual_ipaddress {
    10.239.212.50
}

track_script {
    chk_haproxy
}
}

```

## ANEXO 6: logstash.conf

# Beats -> Logstash -> Elasticsearch pipeline.

```

input {
    beats {
        port => 5000
    }
}

filter {
    grok {
        match => [
            "message" , "%{NOTSPACE:xforw_ip} %{IPV4:src_ip} %{NOTSPACE:id_user} %{NOTSPACE:user} \[%{
            HTTPDATE:logdate}\] \[%{WORD:http_method} %{URI:request_url} HTTP/%{NOTSPACE:protocol_version}\]" %
            {NUMBER:http_status_code}          %{NUMBER:reply_size_include_header}          %{WORD:squid_status}:%
            {WORD:squid_hierarchy}",
            "message" , "%{NOTSPACE:xforw_ip} %{IPV4:src_ip} %{NOTSPACE:id_user} %{NOTSPACE:user} \[%{
            HTTPDATE:logdate}\] \[%{WORD:http_method} %{URIHOST:request_url} HTTP/%{NOTSPACE:protocol_version}\]" %
            {NUMBER:http_status_code}          %{NUMBER:reply_size_include_header}          %{WORD:squid_status}:%
            {WORD:squid_hierarchy}"
        ]
    }
}

output {
    elasticsearch {
        hosts => [ "elkserver.dap.es:9200" ]
        index => "squid-access"
    }
}

```

## ANEXO 7: Visualizaciones y Dashboard de Kibana

```

[
  {
    "_id": "837dc7c0-5176-11e9-bb5a-3b619b54d72e",
    "_type": "dashboard",
    "_source": {
      "title": "Panel de Control",
      "hits": 0,
      "description": "",
      "panelsJSON": "[{\"embeddableConfig\":{},\"gridData\":{\"x\":0,\"y\":0,\"w\":13,\"h\":15,\"i\":\"1\"},\"id\":\"0e7cf240-516f-11e9-bb5a-3b619b54d72e\", \"panelIndex\":\"1\", \"type\":\"visualization\", \"version\":\"6.7.0\"}, {\"embeddableConfig\": {}, \"gridData\": {\"x\":0,\"y\":30,\"w\":48,\"h\":13,\"i\":\"2\"}, \"id\":\"bd91dca0-5174-11e9-bb5a-3b619b54d72e\", \"panelIndex\":\"2\", \"type\":\"visualization\", \"version\":\"6.7.0\"}, {\"embeddableConfig\": {}, \"gridData\": {\"x\":13,\"y\":0,\"w\":11,\"h\":15,\"i\":\"3\"}, \"id\":\"0e605ad0-5175-11e9-bb5a-3b619b54d72e\", \"panelIndex\":\"3\", \"type\":\"visualization\", \"version\":\"6.7.0\"}, {\"embeddableConfig\": {}, \"gridData\": {\"x\":24,\"y\":15,\"w\":24,\"h\":15,\"i\":\"4\"}, \"id\":\"3d322230-5175-11e9-bb5a-3b619b54d72e\", \"panelIndex\":\"4\", \"type\":\"visualization\", \"version\":\"6.7.0\"}, {\"embeddableConfig\": {}, \"gridData\": {\"x\":0,\"y\":15,\"w\":24,\"h\":15,\"i\":\"5\"}, \"id\":\"6b1b5e50-5175-11e9-bb5a-3b619b54d72e\", \"panelIndex\":\"5\", \"type\":\"visualization\", \"version\":\"6.7.0\"}, {\"embeddableConfig\": {}, \"gridData\": {\"x\":24,\"y\":0,\"w\":12,\"h\":15,\"i\":\"6\"}, \"id\":\"Top-10-Mas-accedidos\", \"panelIndex\":\"6\", \"type\":\"visualization\", \"version\":\"6.7.0\"}, {\"gridData\": {\"x\":36,\"y\":0,\"w\":12,\"h\":15,\"i\":\"7\"}, \"version\":\"6.7.0\", \"panelIndex\":\"7\", \"type\":\"visualization\", \"id\":\"4481ae40-520e-11e9-bb5a-3b619b54d72e\", \"embeddableConfig\": {}}]",
      "optionsJSON": "{\"darkTheme\":false,\"hidePanelTitles\":false,\"useMargins\":true}",
      "version": 1,
    }
  }
]

```

```

"timeRestore": false,
"kibanaSavedObjectMeta": {
  "searchSourceJSON": "{\"query\":{\"language\":\"lucene\",\"query\":\"\",\"filter\":[]}}\"
}
},
{
  "_id": "0e605ad0-5175-11e9-bb5a-3b619b54d72e",
  "_type": "visualization",
  "_source": {
    "title": "HTTP_STATUS_CODE",
    "visState": {
      "title": "HTTP_STATUS_CODE", "type": "pie", "params": {
        "type": "pie", "addTooltip": true, "addLegend": true, "legendPosition": "right", "isDonut": true, "labels": {
          "show": false, "values": true, "last_level": true, "truncate": 100, "aggs": {
            [{"id": "1", "enabled": true, "type": "count", "schema": "metric", "params": {}},
            [{"id": "2", "enabled": true, "type": "terms", "schema": "segment", "params": {
              "field": "http_status_code.keyword", "size": 5, "order": "desc", "orderBy": "_key", "otherBucket": false, "otherBucketLabel": "Other", "missingBucket": false, "missingBucketLabel": "Missing", "exclude": ""}]}],
          "uiStateJSON": "{}",
          "description": "",
          "version": 1,
          "kibanaSavedObjectMeta": {
            "searchSourceJSON": "{\"index\":\"e7540fb0-5145-11e9-bb5a-3b619b54d72e\",\"query\":{\"query\":\"\",\"language\":\"lucene\",\"filter\":[]}}\"
          }
        }
      },
    },
    {
      "_id": "6b1b5e50-5175-11e9-bb5a-3b619b54d72e",
      "_type": "visualization",
      "_source": {
        "title": "SQUID_REQUEST_STATUS",
        "visState": {
          "title": "SQUID_REQUEST_STATUS", "type": "pie", "params": {
            "type": "pie", "addTooltip": true, "addLegend": true, "legendPosition": "right", "isDonut": true, "labels": {
              "show": false, "values": true, "last_level": true, "truncate": 100, "aggs": {
                [{"id": "1", "enabled": true, "type": "count", "schema": "metric", "params": {}},
                [{"id": "2", "enabled": true, "type": "terms", "schema": "segment", "params": {
                  "field": "squid_status.keyword", "size": 5, "order": "desc", "orderBy": "1", "otherBucket": false, "otherBucketLabel": "Other", "missingBucket": false, "missingBucketLabel": "Missing"}]}],
                "uiStateJSON": "{}",
                "description": "",
                "version": 1,
                "kibanaSavedObjectMeta": {
                  "searchSourceJSON": "{\"index\":\"e7540fb0-5145-11e9-bb5a-3b619b54d72e\",\"query\":{\"query\":\"\",\"language\":\"lucene\",\"filter\":[]}}\"
                }
              },
            },
            {
              "_id": "3d322230-5175-11e9-bb5a-3b619b54d72e",
              "_type": "visualization",
              "_source": {
                "title": "HTTP_METHOD",
                "visState": {
                  "title": "HTTP_METHOD", "type": "pie", "params": {
                    "type": "pie", "addTooltip": true, "addLegend": true, "legendPosition": "right", "isDonut": true, "labels": {
                      "show": false, "values": true, "last_level": true, "truncate": 100, "aggs": {
                        [{"id": "1", "enabled": true, "type": "count", "schema": "metric", "params": {}},
                        [{"id": "2", "enabled": true, "type": "terms", "schema": "segment", "params": {
                          "field": "http_method.keyword", "size": 5, "order": "desc", "orderBy": "1", "otherBucket": false, "otherBucketLabel": "Other", "missingBucket": false, "missingBucketLabel": "Missing"}]}],
                        "uiStateJSON": "{}",
                        "description": "",
                        "version": 1,
                        "kibanaSavedObjectMeta": {
                          "searchSourceJSON": "{\"index\":\"e7540fb0-5145-11e9-bb5a-3b619b54d72e\",\"query\":{\"query\":\"\",\"language\":\"lucene\",\"filter\":[]}}\"
                        }
                      },
                    },
                    {
                      "_id": "0e7cf240-516f-11e9-bb5a-3b619b54d72e",
                      "_type": "visualization",
                      "_source": {
                        "title": "Peticones por Proxy",
                        "visState": {"title": "Peticones por Proxy", "type": "histogram", "params": {"type": "histogram", "grid": {
                          "categoryLines": false, "style": {"color": "#eee"}, "categoryAxes": [{"id": "CategoryAxis-

```

```

1\,"type":"category",\position":"bottom",\show":true,\style":{\},\scale":{\type":"linear"},\labels":
{\show":true,\truncate":100},\title":{\}},\valueAxes":[{\id":"ValueAxis-1",\name":"LeftAxis-
1",\type":"value",\position":"left",\show":true,\style":{\},\scale":{\type":"linear",\mode":"normal"},\labels":
{\show":true,\rotate":0,\filter":false,\truncate":100},\title":{\text":"Total
peticiones"}},\seriesParams":
[{\show":true,\type":"histogram",\mode":"stacked",\data":{\label":"Total
peticiones",\id":"1"},\valueAxis":"ValueAxis-
1",\drawLinesBetweenPoints":true,\showCircles":true}],\addTooltip":true,\addLegend":true,\legendPosition":"right",
\times":[],\addTimeMarker":false},\aggs":
[{\id":"1",\enabled":true,\type":"count",\schema":"metric",\params":{\customLabel":"Total
peticiones"}},
{\id":"2",\enabled":true,\type":"terms",\schema":"segment",\params":
{\field":"host.name.keyword",\size":10,\order":"desc",\orderBy":"1",\otherBucket":false,\otherBucketLabel":"
Other",\missingBucket":false,\missingBucketLabel":"Missing",\customLabel":"Peticones por Proxy"}]},
"uiStateJSON": "{}",
"description": "",
"version": 1,
"kibanaSavedObjectMeta": {
  "searchSourceJSON": {"index":"e7540fb0-5145-11e9-bb5a-3b619b54d72e",\query":
{\query":"",\language":"lucene",\filter":[]}}
}
},
{
  "_id": "Top-10-Mas-accedidos",
  "_type": "visualization",
  "_source": {
    "title": "Top 10 Usuarios",
    "visState": {"title":"Top 10
Usuarios",\type":"pie",\params":
{\shareYAxis":true,\addTooltip":true,\addLegend":true,\isDonut":false,\legendPosition":"right",\type":"pie",\lab
els":{\show":false,\values":true,\last_level":true,\truncate":100},\aggs":
[{\id":"1",\enabled":true,\type":"count",\schema":"metric",\params":{\}},
{\id":"2",\enabled":false,\type":"filters",\schema":"split",\params":{\filters":{\input":{\query":
{\query_string":{\query":"type:
'squid",\analyze_wildcard":true}}},\row":true}},
{\id":"3",\enabled":true,\type":"terms",\schema":"segment",\params":
{\field":"user.keyword",\size":10,\order":"desc",\orderBy":"1",\otherBucket":false,\otherBucketLabel":"Other",
,\missingBucket":false,\missingBucketLabel":"Missing",\exclude":"","\customLabel":"TOP
10 Mas
accedidos"}]},
"uiStateJSON": "{}",
"description": "",
"version": 1,
"kibanaSavedObjectMeta": {
  "searchSourceJSON": {"index":"e7540fb0-5145-11e9-bb5a-3b619b54d72e",\query":{\query":
{\query_string":{\query":"","\analyze_wildcard":true,\default_field":"","\language":"lucene",\filter":[]}}
}
}
},
{
  "_id": "4481ae40-520e-11e9-bb5a-3b619b54d72e",
  "_type": "visualization",
  "_source": {
    "title": "Top 10 Web Solicitadas",
    "visState": {"title":"Top 10 Web
Solicitadas",\type":"pie",\params":
{\shareYAxis":true,\addTooltip":true,\addLegend":true,\isDonut":false,\legendPosition":"right",\type":"pie",\lab
els":{\show":false,\values":true,\last_level":true,\truncate":100},\aggs":
[{\id":"1",\enabled":true,\type":"count",\schema":"metric",\params":{\}},
{\id":"2",\enabled":false,\type":"filters",\schema":"split",\params":{\filters":{\input":{\query":
{\query_string":{\query":"type:
'squid",\analyze_wildcard":true}}},\row":true}},
{\id":"3",\enabled":true,\type":"terms",\schema":"segment",\params":
{\field":"request_url.keyword",\size":10,\order":"desc",\orderBy":"1",\otherBucket":false,\otherBucketLabel":"
Other",\missingBucket":false,\missingBucketLabel":"Missing",\exclude":"","\customLabel":"TOP
10 Mas
accedidos"}]},
"uiStateJSON": "{}",
"description": "",
"version": 1,
"kibanaSavedObjectMeta": {
  "searchSourceJSON": {"index":"e7540fb0-5145-11e9-bb5a-3b619b54d72e",\query":{\query":
{\query_string":{\query":"","\analyze_wildcard":true,\default_field":"","\language":"lucene",\filter":[]}}
}
}
}
}
]

```

## ANEXO 8: Manual Operacional de la solución.

A continuación se indican un listado de operaciones básicas para el correcto mantenimiento del sistema.

Arranque y parada de servicios:

- Squid  
`/etc/init.d/squid46 {start/stop/restart/status}`
- Clamav - motor de antivirus  
`/etc/init.d/clamav-freshclam {start/stop/restart/status}`
- Servicio c-icap  
`/etc/init.d/c-icap {start/stop/restart/status}`
- Servicio apache2  
`/etc/init.d/apache2 {start|stop|graceful-stop|restart|reload|force-reload}`
- Servicio Filebeat  
`/etc/init.d/filebeat {start|stop|status|restart|force-reload}`
- Servicio Elasticsearch  
`/etc/init.d/elasticsearch {start|stop|restart|force-reload|status}`
- Servicio Logstash  
`systemctl {start/stop/status} logstash`
- Servicio Kibana  
`systemctl {start/stop/status} kibana`
- Servicio Keepalived  
`/etc/init.d/keepalived {start|stop|restart|reload|force-reload}`
- Servicio Haproxy  
`/etc/init.d/haproxy {start|stop|reload|restart|status}`

Inclusión de dominios en lista de ACL.

- Modificación de fichero de acl e introducir dominio en formato “.dominio.dom” o “dominio.dom” o “servicio.dominio.dom”.
- Relectura de los ficheros de configuración  
`/opt/squid4/sbin/squid -k reconfigure`

Limpieza de cache del proxy.

- Parada del servicio squid
- Borrado de las carpetas donde se almacenan la cache (`/var/spool/squid`).
- Recrear la cache  
`/opt/squid46/sbin/squid -z`
- Arranque del servicio squid.

Habilitar el modo debug de squid.

- Añadir el siguiente parametro que volcará toda la información posible del funcionamiento de squid.  
`debug_options ALL,1 33,2 28,9`
- En el caso de un “falso positivo” de una dirección web, puede buscarse en los ficheros de log qué ACL activa el bloqueo gracias al parámetro anteriormente indicado.
- El uso de esta configuración debe realizarse de forma controlada ya que generará una enorme cantidad de información por cada una de las peticiones que se le realice.