Capítulo 2

Escenario y objetivos

En este capítulo se describe el escenario donde se desarrolla el proyecto. Se hará en primer lugar una descripción general de la aplicación sobre la que hemos trabajado con sus características, posteriormente se describirán los subsistemas que la componen, y por último se dará una visión general del funcionamiento de esta aplicación de partida.

Una vez vista la aplicación de partidad se definirán los objetivos marcados en el proyecto.

2.1. Descripción general

Este proyecto está basado en un sistema que desarrollan actualmente una comunidad en el condado de Sonoma en Estados Unidos, el proyecto que realizan lo denominan *The NoCat Community Wireless Network Project*. La última versión publicada ha sido la 0.82 y es sobre la cual hemos desarrollado nuestra ampliación. El proyecto NoCat tiene como objetivo desarrollar una aplicación que permita controlar el acceso a una red (normalmente Internet) desde una red local. Para ello consta de dos partes claramente diferenciadas: un equipo que interconecta las dos redes, en adelante lo llamaremos *pasarela*; y un servidor de autenticación donde se guarda la información de los usuarios.

Lo que hace este sistema básicamente es que cuando un usuario intenta acceder fuera de su red local a través de la pasarela mediante un navegador web, se le muestra una pantalla para que inserte su login y su clave, se contacta con el servidor, se comprueba los permisos del usuario, y la pasarela se encarga de permitir o no su acceso. Además la pasarela también reserva una calidad de servicio según unas clases predefinidas.

Las clases que tiene NoCat son tres:

- Clase "Public". A la clase "Public" pertenecen los usuarios que están en la base de datos del servidor de autenticación pero no pertenecen a ningún grupo. Además al presentar la pantalla con el login se da la oportunidad de saltar esa página y navegar en la clase "Public".
- Clase "Member". A esta clase pertenecen los usuarios que están en la base de datos del servidor y además pertenecen a algún grupo.
- Clase "Owner". Reservada para grupos predefinidos que suelen ser los propietarios de la pasarela o los que proveen el servicio.

Un ejemplo de la topología que se va tener se puede ver en la figura 2.1.

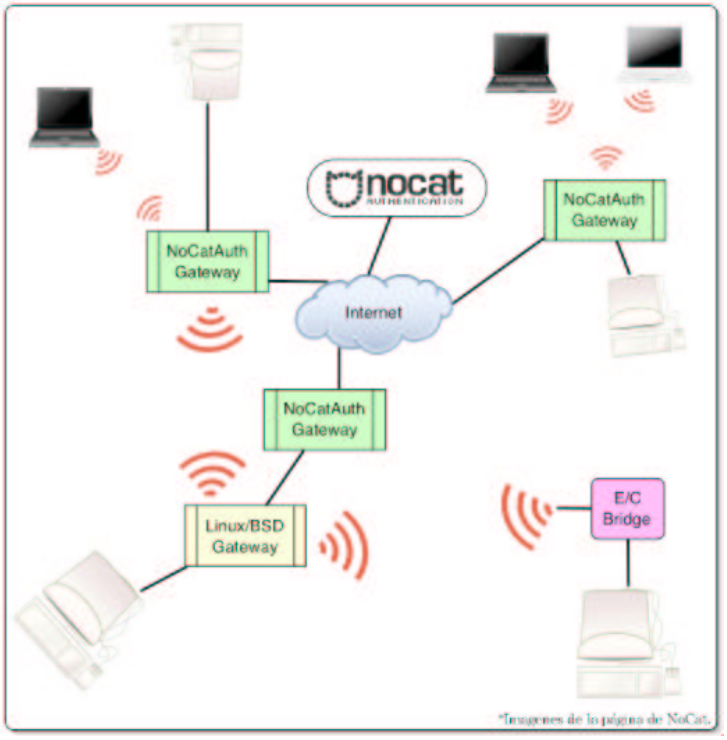


Figura 2.1: Topología de la red

2.2. Descripción de subsistemas

2.2.1. Pasarela

La pasarela será un equipo de la red a través del cual se acceda a Internet. Este equipo intermedio tiene capacidad para supervisar todas las comunicaciones y actuar

sobre ellas en base a órdenes recibidas del servidor de autenticación.

El método que utiliza la pasarela para controlar los accesos es muy simple, pero muy eficaz. Se trata de una máquina Linux 2.4.x con iptables[1], además necesita gpgv que sirve para comprobar la autenticidad del servidor y cifrar la comunicación.

Mediante las iptables la pasarela marcará los paquetes según su dirección IP origen y su dirección IP destino, asimismo también tendrá en cuenta la MAC origen, y la MAC destino. Las marcas que usa la pasarela son cuatro: 0x1, 0x2, 0x3 y 0x4, y la asignación que realiza a cada marca es: clase Owner, clase Member, clase Public y "sin clase" respectivamente.

2.2.2. Servidor de autenticación

El servidor de autenticación es un equipo central que contiene los usuarios y el grupo al que pertenece cada usuario. Este servidor puede ser accedido por varias pasarelas a la vez y almacenar la información de muchos usuarios. Por tanto estamos hablando de un sistema centralizado, y como tal, el servidor de autenticación debe ser una máquina con mucha capacidad para que no se colapse son facilidad.

El servidor de autenticación debe cumplir los siguientes requisitos:

- Tener instalado un servidor web seguro que soporte el uso de protocolos SSL[2] y TLS[3].
- Tener un interprete de Perl 5 (5.6 o mejor recomendado).
- Tener los siguientes módulos de perl: Digest::MD5, DBI y DBD::MySQL.
- Gnu Privacy Guard(gnupg). Para cifrar las comunicaciones entre la pasarela y el servidor.

En el servidor hay que crear una base de datos que se llamará nocat que incluye dos tablas: la tabla *member* y la tabla *network*. En la tabla *member* se guarda el login y la clave de cada usuario, mientras que en la tabla *network* se guarda el grupo al que pertenecen los usuarios. Además de estas tablas se incluyen algunas más, pero en la versión 0.82, que es la última aún no se utilizan esas tablas.

Para realizar la autenticación, en el servidor se ejecutan varios scripts en cgi. El principal es login.cgi que veremos con más detalle en el apéndice A.1.

[1] Ver el apéndice A.3 para conocer el funcionamiento de iptables.
[2] Secure Sockets Layer.
[3] Transport Layer Security.

2.3. Proceso de funcionamiento

Vamos a dividir el proceso en varias fases:

1. Cuando se inicia la pasarela se ejecuta un programa que inicia las reglas del cortafuegos de la siguiente manera: marca todos los paquetes que provengan de la red local con la marca 0x4 y a su vez redirige todos los paquetes con la marca 0x4 al servidor de autenticación. Además crea tres reglas para las marcas 0x1, 0x2 y 0x3 y los permite navegar, pero en principio no crea ninguna regla para marcar paquetes con esas marcas.

2. Cuando un miembro de la red local intenta navegar por Internet a través de la pasarela, las reglas que se han definido al inicio marcan el paquete con la marca 0x4 y esta petición es capturada por la pasarela. La pasarela realiza algunas comrobaciones y se redirige al usuario a la página del servidor de autenticación que muestra un formulario.

3. El usuario rellena el formulario con el login y la clave y envía esta información. El servidor comprueba que ese login y clave corresponden con alguno en su tabla *member* y además comprueba si el usuario pertenece a algún grupo. A partir de aquí pueden ocurrir dos cosas:

   a) Puede ser que el usuario no sea autenticado o haya algún error, en ese caso no se permite navegar al usuario y se muestra un mensaje de error.

   b) Si el usuario ha sido autenticado se desencadena un proceso para comunicárselo a la pasarela:

      1) El servidor devuelve al usuario una página que hace lo siguiente:
         - Por un lado lleva un mensaje encriptado con los datos de usuario y la clase a la que pertenece. Este mensaje es reenviado del usuario a la pasarela para decirle a la pasarela que debe permitir el acceso a la red.
         - Por otro lado, la misma información que contiene el mensaje que se enviará a la pasarela se guarda en un popup[4] que le salta al usuario.

      2) Una vez que la pasarela recibe el mensaje encriptado del usuario lo desencripta con la clave pública y crea una regla nueva en el cortafuegos. Esta regla consiste en marcar los paquetes con origen la dirección IP y MAC del usuario en cuestión. La marca que se utiliza es la correspondiente a su clase (0x1 si es de la clase Owner, 0x2 si es de la clase Member y 0x3 si es de la clase Public). A partir de este momento el usuario podrá navegar ya que al iniciar la pasarela se habían creado reglas para permitir al acceso a los paquetes marcados con esas marcas.

[4]Página que salta de forma automática en una ventana nueva.

3) En la pasarela se puede definir un parámetro que es tiempo de conexión que tiene un usuario una vez que ha sido autenticado. Si expira este tiempo automáticamente se borra la regla que hay en el cortafuegos del marcado de sus paquetes, por lo que sus paquetes volverían a ser marcados con la marca 0x4 y serían redirigidos hacia el servidor de autenticación. Para que el usuario no tenga que estar autenticándose cada cierto tiempo cuando expire su temporizador está el popup del que hemos hablado antes.

4) El popup está programado para que cuando haya transcurrido una fracción[5] del tiempo de expiración mande la información que contiene al servidor de autenticación. Entonces el servidor de autenticación le devuelve otra vez un mensaje cifrado para que lo mande a la pasarela y vuelve a programar el popup para autenticarse antes de que transcurra el tiempo de expiración. Con este método la pasarela recibe un mensaje con las credenciales del usuario antes de que expire su temporizador sin necesidad de molestar la usuario.

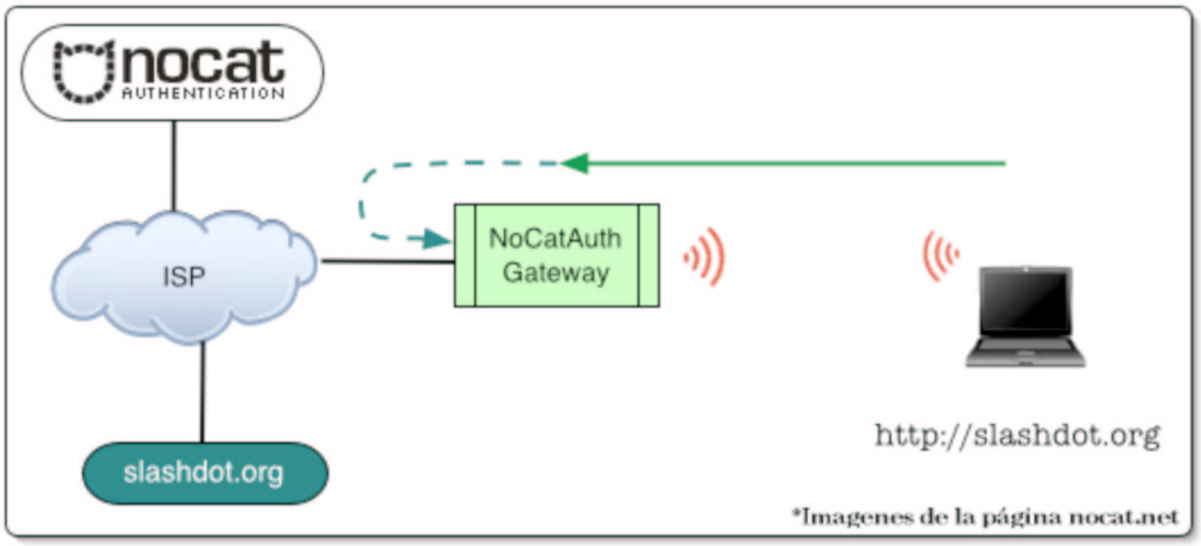
5) La conexión se termina en el momento que el usuario cierra el popup y expira su tiempo, o bien, si el usuario pulsa un botón que trae el popup para salir. Este botón envía la información directamente a la pasarela para borrar la regla correspondiente del cortafuegos.

Básicamente este es el funcionamiento del sistema cuando está actuando en modo Pasivo, es decir, cuando estamos tras una pasarela que actúa como NAT[6]. Para más detalles de la implementación y de las reglas del cortafuegos que se aplican véase el apéndice A.1.

Tanto el demonio que se está ejecutando en la pasarela como los scripts CGI que se ejecutan en el servidor están escritos en Perl.

[5]Concretamente el valor de la fracción es 0.75.

[6]Network Address Translation.



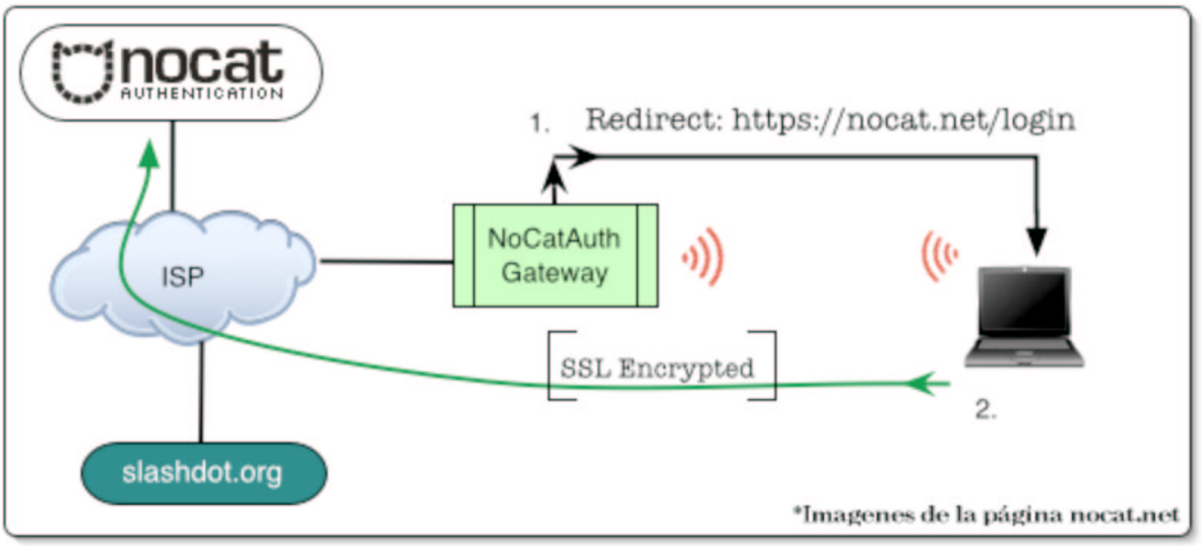
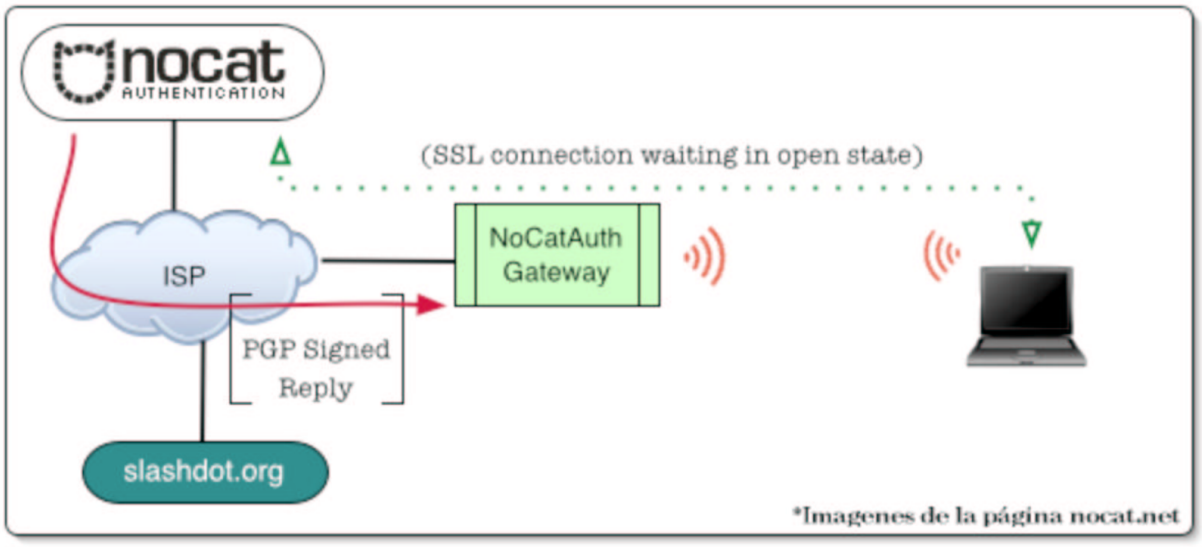
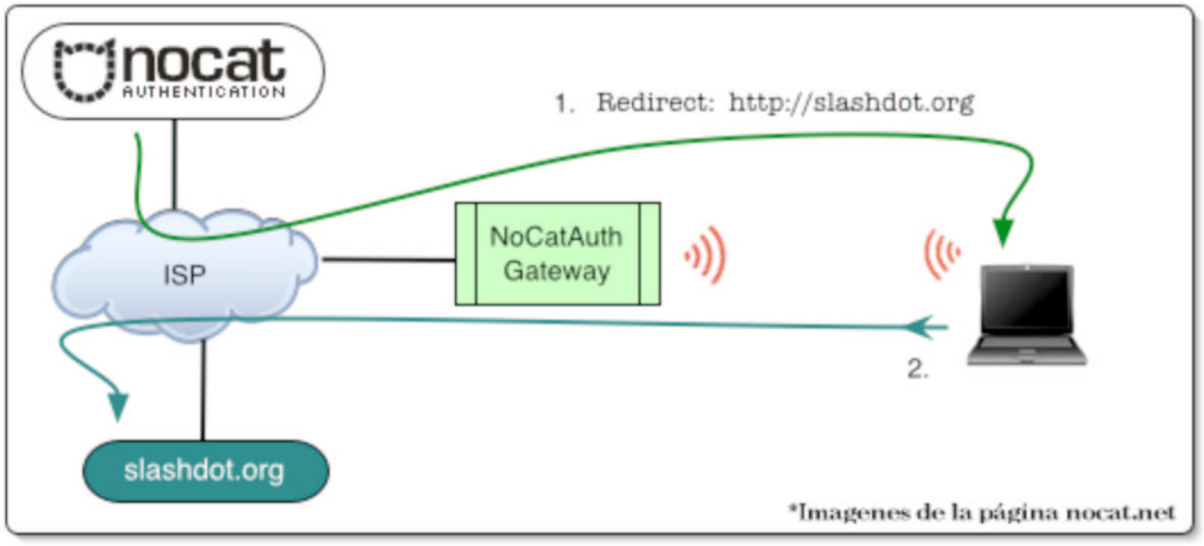


Figura 2.2: Funcionamiento de NoCatAuth



Fase 3: Notificación a la pasarela



Fase 4: Redirección a la página requerida

Figura 2.3: Funcionamiento de NoCatAuth

2.4. Objetivos

Los objetivos de este proyecto vienen marcados por las necesidades de la empresa Eneo Tecnología, como ya se ha dicho en la introducción. En este capítulo se desarrollan un poco más las ideas que se han comentado en el capítulo de introducción.

- El primer objetivo de este proyecto es realizar un sistema que funcione. Es decir, buscamos desarrollar una primera versión del sistema, con una funcionalidad básica, pero que sirva para poner en marcha el negocio. La aplicación deberá cumplir una serie de requisitos que comentaremos más adelante, pero debido a la escasez de tiempo del que dispone la empresa, se pretende dejar abierto el proyecto para posteriores mejoras.
- Control de acceso. NoCatAuth ofrece un control de acceso muy rudimentario, la única funcionalidad que tiene implementada es comprobar la existencia de un usuario en la base de datos y permitirle el paso. Nosotros pretendemos ampliar esta funcionalidad a:
  - El sistema debe registrar el momento de entrada y salida de un usuario.
  - El sistema debe ser capaz de expulsar a un usuario de la red en el momento que agota el servicio contratado.
  - Comprobar en el momento de conexión que un usuario tiene contratado algún servicio, y además que no hay nadie dentro del sistema usando ese login.

  Además de realizar estas funciones, nos proponemos que el control de acceso sea fiable, seguro, estable, y sobre todo se va a intentar que sea modular y adaptable a posteriores versiones del nocat. Esto es muy importante ya que el proyecto NoCatAuth sigue avanzando, y cuánto más independiente sea nuestro trabajo de la aplicación NoCatAuth mejor, ya que seguirá funcionando perfectamente con versiones posteriores a la 0.82.
- Calidad de servicio. La calidad de servicio es un tema importante dentro del proyecto, ya que la tarificación se va a realizar por tiempo consumido. No sería muy adecuado cobrar a alguien por el tiempo que ha estado conectado, si durante ese tiempo la red ha estado congestionada y apenas ha dispuesto de ancho de banda.

  El objetivo a conseguir es que los parámetros de calidad de servicio sean:
  - Flexibles. Queremos que se puedan configurar unos parámetros de calidad de servicio distintos para cada usuario.
  - Configurables. Se pretende definir un caudal mínimo de bajada, un caudal máximo alcanzable y un parámetro de prioridad para cada usuario. Por ahora el caudal de subida será el mismo para todos los usuarios.

    - Sencilla. Buscamos una implementación sencilla para conseguir una calidad de servicio garantizada.

- Políticas de acceso. Con diferentes políticas de acceso pretendemos ofrecer distintos servicios a distintos precios. Habrá políticas para acceder sólo a tráfico web, otra que te permitan recibir y mandar correo, que dispongas de Telnet, de ftp o de ssh.

- Administración del sistema. El objetivo es permitir que cualquier usuario pueda administrar el sistema desarrollado, sin necesidad de conocer nada del funcionamiento interno. No debe saber que hay una base de datos con unas tablas, etc... El programa de administración debe ser : sencillo, manejable y intuitivo.

  Además de incluir estas ventajas para el usuario, el programa de administración debe ser:

    - Seguro. La aplicación debe estar protegida para que sólo la pueda usar el personal autorizado.
    - Fiable. La aplicación debe mantener la consistencia dentro de la base de datos. La aplicación se encargará de comprobar la validez de datos introducidos.
    - Funcional. La aplicación desarrollada incluirá una serie de funciones básicas de administración, que podrán ser ampliadas en un futuro.