

RADIUS en Linux y Cisco

Emilio José Mira Alfaro

emial@alumni.uv.es 12 de Agosto de 2001

1 Introducción

RADIUS (Remote Authentication Dial In User Service) es un protocolo de control de accesos desarrollado por Livingston Enterprises y que la IETF ha recogido en los RFCs 2865 y 2866. Fue diseñado para autenticar usuarios y utiliza una arquitectura cliente/servidor. El servidor contiene información de los usuarios, almacenando sus passwords y sus perfiles, y el cliente es el encargado de pasar las peticiones de conexión de los usuarios al servidor para que éste las autentique y responda al cliente diciéndole si ese usuario está o no registrado.

Un ejemplo de uso de RADIUS se puede dar en un ISP, donde el NAS (Network Access Server) hace de cliente RADIUS y un host del ISP podría hacer de servidor. El NAS recibe vía módem una petición de acceso y envía los datos que el usuario ha proporcionado al servidor RADIUS. Es éste el que consulta su base de datos y comprueba si el usuario que ha realizado la llamada es en realidad quien dice ser. En ese caso, responde al NAS con una respuesta de aceptación de la llamada y, opcionalmente, con datos sobre el perfil del usuario (tipo de servicio, protocolo de conexión, IP asignada, ...). En caso contrario notifica al NAS que debe rechazar la petición de conexión. Opcionalmente, el servidor RADIUS guardará logs de las conexiones en las que estemos interesados.

A lo largo del texto veremos con algo más de detalle como funciona este protocolo y algunos ejemplos de configuración para autenticar usuarios en Linux y Cisco. El servidor RADIUS elegido para las pruebas ha sido FreeRADIUS y se puede descargar de www.freeradius.org. Los clientes son un PC Linux con PAM v.075 (www.kernel.org/pub/linux/libs/pam/) modificado y un router Cisco 2500.

2 Protocolo RADIUS

La primeras páginas del RFC 2865 (las secciones 1 y 2, páginas de la 1 a la 13) describen el funcionamiento de este protocolo. Ver [4].

3 Clientes RADIUS

3.1 Linux-PAM (Pluggable Authentication Modules for Linux)

Linux-PAM es el sistema de autenticación utilizado en Linux. Antes, cuando una aplicación quería ofrecer un servicio para el que se requería autenticación (como es el caso de login), la aplicación pedía al usuario que tecleara su login y su password, consultaba el fichero `/etc/passwd` para comprobar que los passwords encriptados correspondían, y lanzaba una shell con el PID del usuario. Nuevas formas de autenticación obligaban a nuevos programas login que las implementaran.

Con Linux-PAM se define una interfaz para que los programas como login que requieran de una autenticación sean transparentes al tipo de autenticación utilizada. Para ello se escribió una librería (`libpam.so` y `libpam_misc.so`) que implementaba la interfaz para todo lo relacionado con accesos al sistema. Ahora una aplicación ya no trabaja con el fichero `/etc/passwd`, sino que llama a una función genérica `autenticar(usuario, token)` implementada en la librería y, dependiendo de la configuración que hizo el administrador del sistema, esta función usará una forma u otra de validar a ese usuario.

PAM tiene cuatro formas de actuación, que se transforman en cuatro grupos de funciones distintas dentro de la librería:

- Authentication: utilizada para validar usuarios. Las funciones asociadas son: `int`

```
pam_sm_authenticacion(...) int pam_sm_setcred(...)
```

- Accounting: se utiliza para gestionar la cuenta del usuario (habilitación/deshabilitación de la cuenta, fecha de expiración, etc.). La función asociada es: `int pam_sm_acct_mgmt(...)`
- Password: actualización del password del cliente. Su función es: `int pam_sm_chauthtok(...)`
- Session: gestión de la sesión del usuario (logs al inicio y al final, montaje de directorios del usuario, etc.). Las funciones son: `int pam_sm_open_session(...)` `int pam_sm_close_session(...)`

La aplicación llamará a estas funciones con los parámetros necesarios y PAM hará el resto. El administrador del sistema es el encargado de decirle a PAM como ha de comportarse dependiendo de la aplicación que lo llame. Así, login puede usar una forma de autenticación y xdm puede usar otra distinta, por ejemplo. Para ello aparecen los módulos, que son los encargados de implementar la forma en al que se hará la autenticación, el accounting, la gestión de passwords y el inicio y fin de la sesión. Podemos utilizar un módulo para la gestión de accesos en la forma tradicional que UNIX lo hace y asociárselo a login, y otro modulo distinto que lea el password de una tarjeta electrónica y asignárselo a xdm, por ejemplo.

3.1.1 Configuración de PAM

La configuración de PAM se puede hacer de dos formas distintas: editando el fichero `/etc/pam.conf` o editando los fichero que aparecen en el directorio `/etc/pam.d/`. Si el directorio `/etc/pam.d/` existe, PAM tomará la configuración de los ficheros que contenga, y si no existe, la tomará del fichero `/etc/pam.conf`.

```
/etc/pam.conf
```

Este fichero está formado por líneas sucesivas que siguen la siguiente sintaxis:

```
aplicación tipo-del-modulo control ruta-del-modulo argumentos
```

aplicación

Nombre de la aplicación a la que se le aplicará la configuración que sigue la línea. Ejemplos: login, su, xdm, ...

tipo-del-modulo

Hay cuatro posibles tipos que corresponden con las cuatro formas de actuación que vimos antes: auth, account, password y session.

control

Nos dice como reacciona PAM ante el fallo o éxito de esta línea. Puede tomar cuatro valores distintos:

- required: es necesario que tenga éxito esta línea para que PAM devuelva PAM_SUCCESS a la aplicación. Si falla esta línea, el control no se pasa directamente a la aplicación, sino que se sigue la ejecución del resto de líneas asociadas al mismo tipo-de-modulo, aunque PAM ya sepa que devolverá un fallo a la aplicación.
- requisite: igual que antes, pero ahora, cuando falle esta línea, el control será pasado directamente a la aplicación, sin ejecutar el resto de líneas asociadas a este mismo tipo-de-modulo.
- sufficient: el éxito de esta línea significa el éxito del módulo dentro de este tipo-de-modulo, sin importar las líneas sucesivas de este mismo tipo.
- optional: si tiene o no éxito esta línea no influye para nada en el comportamiento de PAM.

ruta-del-modulo

Localización absoluta del módulo. En Red Hat se encuentra en el directorio `/lib/security/`, mientras que en el resto de distribuciones se suele encontrar en `/usr/lib/security/`.

argumentos

Lista de argumentos para ejecutar el módulo separados únicamente por espacios en blanco.

El funcionamiento de PAM es el siguiente: cuando se ejecuta una aplicación como login, esta aplicación llamará a las funciones que hemos descrito antes. Primero, login llamará a `pam_sm_authentication(...)`, que está asociada con el tipo de módulo `auth`. La llamada tendrá como argumentos, entre otras cosas, el nombre de usuario, el password y el terminal desde el que se está haciendo la conexión. Esta función le da el control a PAM, que buscará en `/etc/pam.conf` todas las líneas cuyo campo de aplicación sea `login` y su tipo-de-módulo, `auth`. Para cada una de estas líneas, PAM cargará el módulo indicado en ruta-del-módulo y lo ejecutará con los argumentos que haya en argumentos. Dependiendo del contenido del campo `control`, PAM reaccionará de una forma u otra ante el fallo o éxito de este módulo. Cuando `pam_sm_authentication(...)` haya finalizado, login puede proceder de dos formas: si la función ha devuelto éxito, puede continuar la autenticación estableciendo los credenciales de este usuario con `pam_sm_setcred(...)` y seguir con el proceso de conexión. Si no ha tenido éxito mostrará un mensaje por pantalla diciendo: ``Login incorrect''.

Veamos un ejemplo de configuración de `/etc/pam.conf`:

```
login    auth        required    /lib/security/pam_securetty.so
login    auth        sufficient  /lib/security/pam_unix.so likeauth nullok md5 shadow
login    auth        required    /lib/security/pam_deny.so
login    auth        required    /lib/security/pam_nologin.so
login    account     sufficient  /lib/security/pam_unix.so
login    account     required    /lib/security/pam_deny.so
login    password    required    /lib/security/pam_cracklib.so retry=3
login    password    sufficient  /lib/security/pam_unix.so nullok use_authok md5 shadow
login    password    required    /lib/security/pam_deny.so
login    session     required    /lib/security/pam_limits.so
login    session     required    /lib/security/pam_unix.so
login    session     optional    /lib/security/pam_console.so
```

Esta parte de `/etc/pam.conf` configura la seguridad en el programa login. Cuando login llame a `pam_sm_authentication(...)`, se buscarán en este fichero las líneas de configuración para login con el tipo de módulo `auth`. PAM ejecutará primero el módulo `pam_securetty.so`, que comprueba que si es el usuario root el que intenta establecer una conexión, ha de hacerlo desde una terminal incluida en el fichero `/etc/securetty`. Si añadimos en este fichero las terminales de la `tty1` a la `tty6`, obligaremos a que el usuario root, siempre que se registre en el sistema, lo haga desde la terminal física y no por telnet. Eso sí, todavía podríamos llegar a ser superusuarios desde telnet por medio del comando `su`.

Si el módulo `pam_securetty.so` devuelve éxito, PAM pasará a la siguiente línea. Para ello carga el módulo `pam_unix.so` que buscará el password de ese usuario en `/etc/shadow` con encriptación `md5`. Si el usuario y el password corresponden con los que PAM le pasó al módulo, devolverá éxito, por lo que como el contenido del campo `control` era `sufficient`, `pam_sm_authentication(...)` devuelve `PAM_SUCCESS` a login, y login sabe que puede continuar con la conexión. En el caso de que el usuario hubiese equivocado su contraseña o que no fuese el usuario legítimo, este módulo fallaría y PAM continuaría con el siguiente, `pam_deny.so`, que fallaría y provocaría que `pam_sm_authentication(...)` devolviese un error a login.

`/etc/pam.d/`

Este otro tipo de configuración es muy parecido a la anterior, solo que ahora, dentro de este directorio, tenemos un fichero por cada aplicación que requiera de los servicios de PAM. Cada fichero tiene el nombre de la aplicación a la que configura. De esta forma, la sintaxis de cada fichero es igual que la de `/etc/pam.conf`, excepto que desaparece el campo `aplicacion`.

Para una información más detallada sobre PAM (funcionamiento, configuración y programación) véase [1].

3.1.2 PAM como cliente RADIUS

La versión utilizada de PAM ha sido la v0.75, que implementa un módulo cliente RADIUS que por desgracia solo dispone del tipo de módulo session, por lo que tan solo nos servirá para llevar un sistema de logs alternativo a syslog, el demonio de logs de Linux. He modificado este módulo para que incluya la autenticación también por RADIUS y está disponible [aquí](#).

Si utilizamos la distribución de Linux Red Hat, para configurar PAM con autenticación y logs mediante RADIUS el fichero `/etc/pam.d/system-auth` debería quedar así:

```
#auth      sufficient    /lib/security/pam_unix.so likeauth nullok md5 shadow
auth      sufficient    /lib/security/pam_radius.so
auth      required      /lib/security/pam_denial.so
account   sufficient    /lib/security/pam_unix.so
account   required      /lib/security/pam_denial.so
#password  required      /lib/security/pam_cracklib.so retry=3
#password  sufficient    /lib/security/pam_unix.so nullok use_authok md5 shadow
password  required      /lib/security/pam_denial.so
session   required      /lib/security/pam_limits.so
session   required      /lib/security/pam_unix.so
session   required      /lib/security/pam_radius.so
```

Comentamos la primera línea de auth porque ahora la autenticación no se realizará por medio del módulo pam_unix.so, es decir, consultando `/etc/shadow` con encriptación md5 (tal y como indican las opciones que se le pasan al módulo). Comentamos las dos primeras líneas de password porque ahora los passwords residen en el servidor RADIUS, pero no descomentamos la tercera porque así obligamos a que cualquier intento de cambio de password por cualquier aplicación que trabaje con PAM (podemos comprobar que todas acaban llamando al fichero de configuración `system-auth` para el tipo de módulo password) sea denegado. Para que el usuario fuese capaz de modificar su password (algo muy deseable entre otras cosas por motivos de seguridad) habría que implementar en el módulo pam_radius.so la función `pam_sm_chauthtok(...)`, que sería la encargada de comunicarse con el servidor RADIUS y cambiar la contraseña en la base de datos de éste. Esto es fácil si para ello utilizamos la librería `pwddb.a` que ya implementa una función de cambio de password y en la que se basa todo el módulo pam_radius.so.

3.2 Cisco AAA (Authentication, Authorization and Accounting)

Cisco incorpora en sus equipos gestión para autenticación, autorización y accounting (gestión de logs). La forma en la que podemos configurar dispositivos de Cisco para que hagan estas tareas es mediante los comandos AAA.

Veremos estos comandos en un ejemplo de configuración de un router Cisco 2500 como cliente RADIUS.

3.2.1 AAA

Para habilitar AAA deberemos escribir el siguiente comando en modo de configuración global:

```
aaa new-model
```

Las características que veremos a continuación de AAA no estarán disponibles hasta que no emitamos este comando.

El debug de AAA se puede activar con los comandos:

```
debug aaa authentication on
debug aaa authorization on
debug aaa accounting on
```

3.2.2 Authentication

La autenticación permite al dispositivo validar usuarios. Podemos definir listas de métodos (method lists) para indicarle a router los pasos que debe de seguir para realizar esta tarea. Veamos cual es el comando principal de autenticación:

```
aaa authentication type {default | list-name} method1 [method2...]
```

donde

type: puede ser login, ppp y enable, entre otras cosas.

list-name: define el nombre de la lista que luego será aplicada a distintos interfaces o líneas.

method: define el método utilizado para llevar a cabo la autenticación.

La autenticación la podemos realizar sobre varios tipos de passwords mediante el campo type. Esto nos permite asociar una forma de autenticación a las conexiones PPP distinta de la autenticación por Telnet y distinta también a la autenticación del modo privilegiado del router, por ejemplo. A esta forma de autenticación le podemos dar un nombre, para después aplicarla sobre una determinada línea si el campo type era line, o sobre un interfaz serie si era ppp. Si elegimos el nombre default, esta lista será aplicada por defecto a todos los interfaces o líneas (dependiendo del campo type). Por último, la lista de métodos es la que nos dice el orden en el que se van a utilizar los métodos de autenticación. Existen varios tipos de métodos:

Palabra clave	Descripción
enable	Usa el password de enable para autenticar.
krb5	Usa Kerberos 5 para autenticar.
line	Usa el password asociado con la línea para autenticar.
local	Usa la base de datos local para autenticar.
none	No usa autenticación.
radius	Usa RADIUS para autenticar.
tacacs+	Usa TACACS+ para autenticar.
krb5-telnet	Usa Kerberos 5 Telnet autenticación cuando se usa Telnet para conectarse al router.

Veremos a continuación varios ejemplos de configuración de la autenticación con AAA:

Autenticación de login

Para configurar la autenticación de las líneas virtuales y de consola de forma que los passwords se busquen de forma local, en modo de configuración global deberemos teclear lo siguiente:

```
aaa authentication login default local
```

Los passwords los tendremos que haber definido antes mediante el comando:

```
username nombre password passwd
```

Si lo que queremos es que los passwords sean los asociados a cada línea, deberemos usar la siguiente configuración:

```
aaa authentication login default line
```

asegurandonos de configurar los passwords adecuados en cada línea con estos comandos:

```
line [ aux | console | tty | vty ] numero-línea [numero-final-línea]
password passwd
```

También es posible definir varias listas con distintos metodos de autenticación y aplicar cada uno a lineas disferentes. Por ejemplo, vamos a definir dos listas, lista1 y lista2, una que autentique por RADIUS y la otra por TACACS+. La primera la aplicaremos a la línea de consola y la segunda a las líneas virtuales:

```
aaa authentication login conline radius local

aaa authentication login vtyline tacacs+ local

!---- Línea de consola

line con 0

login authentication conline

!---- Líneas virtuales

line vty 0 4

login authentication vtyline
```

Vemos como las dos listas de metodos están formadas por dos tipos de autenticación: radius o tacacs+ y local. Esto significa que si no es posible recibir una respuesta de los servidores RADIUS o TACACS+ porque ha caído el servidor, por ejemplo, se recurrirá a buscar los passwords en la base de datos local. Es importante destacar que esto no significa que si RADIUS o TACACS+ devuelven un mensaje de "acceso denegado" se recurrirá a la base de datos local, sino que se denegará el acceso.

Es necesario decirle al router quién es su servidor RADIUS y cual es su secreto. Para ello teclearemos los comandos:

```
radius-server host 192.168.40.1

radius-server key SecretoRadius
```

Autenticación de PPP

En el caso en el que tengamos un enlace serie que necesite de autenticación, podemos usar la siguiente línea para que sea RADIUS o la base de datos local en caso de fallo de RADIUS, los que autentiquen la conexión:

```
aaa authentication ppp default radius local
```

Para configurar un enlace PPP entre dos routers (Router_A y Router_B) para que utilice PAP y que sea Router_A el que deba registrarse en Router_B, teclearemos la siguiente configuración:

```
Router_A

int serial 0

encapsulation ppp

ppp authentication pap callin

ppp pap sent-username Router_B password passwd

Router_B

int serial 0

encapsulation ppp

ppp authentication pap
```

La sintaxis general del último comando utilizado es la siguiente:

```
ppp authentication {chap | pap | chap pap | pap chap} {default | list-name}
[callin]
```

, donde con chap pap o pap chap le decimos que use el segundo si el primero no está soportado por la otra parte, y con callin le decimos que seremos nosotros los que nos autentiquemos ante la otra parte.

Si lo que queremos es definir una lista diferente para cada interfaz que use PPP, haremos lo siguiente:

```
aaa authentication ppp radius-list radius aaa authentication ppp local-list
local int serial 0 ppp authentication pap radius-list int serial 1 ppp
authentication pap local-list
```

3.2.3 Authorization

Authorization permite configurar el tipo de acceso que un usuario tendrá después de que haya sido autenticado: tipos de conexiones a las que tiene acceso (PPP, SLIP), NASs desde los que se puede conectar, etc. Esta opción aparece en la serie 7200 y no está disponible en los routers 2500 y 2600, por lo que no la trataremos aquí. Se puede encontrar información sobre la autorización en el web de Cisco:

www.cisco.com/univercd/cc/td/doc/product/software/ios113ed/113ed_cr/secur_c/scprt1/scauthor.htm

3.2.4 Accounting

Accounting nos permite llevar un registro de los servicios a los cuales los usuarios están accediendo. El comando general para configurar el accounting es el siguiente:

```
aaa accounting {system | network | connection | exec | comand level} {start-stop
| wait-start | stop-only} {tacacs+ | radius}
```

Cisco soporta 5 tipos diferentes de accounting:

- network: nos informa de todas las sesiones PPP y SLIP incluyendo contadores del número de paquetes y bytes.
- connection: nos informa de todas las conexiones outbound (Telnet, rlogin, ...).
- exec: informa de todas las shells de usuario que son ejecutadas en el NAS.
- system: provee información sobre todos los eventos a nivel de sistema (reboot, apagado del equipo).
- command: nos informa de los comandos para un nivel de privilegio específico.

Dependiendo de la cantidad de información que queramos, elegiremos:

- start-stop: toma cuenta de cuando el usuario inicia y finaliza el servicio.
- wait-start: como antes, pero antes de ofrecer el servicio al usuario se asegura de que el servidor de logs (RADIUS o TACACS+) han tomado nota del registro del NAS.
- stop-only: solo informa de cuando el servicio finaliza.

Por último le especificamos que tipo de servidor tomará nota de los logs que genere.

4 Servidor RADIUS

El servidor RADIUS que vamos a utilizar será FreeRADIUS. Lo podemos descargar de www.freeradius.org en forma de fichero .tar.

La instalación de FreeRADIUS genera un árbol de directorios dentro de /usr/local/etc/raddb/. Los ficheros de configuración más importantes son:

- users: contiene información de acceso para cada usuario, como el password, el tipo de servicio, el NAS al que se puede conectar, etc.
- clients: en este fichero se encuentran los posibles clientes del servidor RADIUS con sus respectivos secretos.
- radiusd.conf: fichero de configuración general del demonio radiusd.
- dictionary: contiene el valor numérico que tiene cada atributo y valor. Esto hace a RADIUS un protocolo extremadamente ampliable, puesto que podemos añadir nuevas funcionalidades con una parte cliente ampliada simplemente editando este fichero.

Para hacer pruebas es conveniente lanzar el demonio RADIUS (radiusd) con la opción `-x`, que nos permitirá ver las conexiones que se realizan así como los logs que generan.

4.1 users

La sintaxis general es la siguiente:

```
nombre_de_usuario    check_item_1, check_item_2, ..., check_item_i
                    reply_item_1,
                    reply_item_2,
                    ...
                    reply_item_j
```

Un check item es un atributo de la forma nombre = valor. Cuando un cliente envía un ACCESS-REQUEST al servidor RADIUS, éste buscará en su fichero users un nombre de usuario que coincida con el de la petición. Una vez encontrado, si todos los check items que contiene este usuario coinciden con los que vinieron en la petición del cliente, la conexión será aceptada, es decir, el servidor RADIUS enviará un ACCESS-ACCEPT [4]. En esta respuesta incluirá también los reply items, que usará el cliente para configurar la conexión del usuario, dándole por ejemplo una dirección IP, estableciendo un filtro determinado, la MTU, etc. Si no se hubiese cumplido uno o varios check items, el servidor RADIUS habría respondido con un mensaje ACCESS-REJECT, y el cliente habría negado la conexión al usuario.

Veamos ahora varios ejemplos para configurar accesos a un router Cisco que usa AAA mediante RADIUS. (Al no ser posible la autorización en los equipos de Cisco, cualquier restricción de acceso ha de hacerse desde los check items, por lo que el uso de reply items no tiene sentido en nuestro caso. Si los equipos Cisco hubiesen soportado autorización, en los check items tan solo habrían sido necesarios los atributos Auth-Type y Password, pasando el resto de atributos a la parte de reply items.) . Para más información consultar [3].

Acceso por Telnet `ciscovty Auth-Type:=Local, Password == ``passvty'', NAS-Port-Type == Virtual, Calling-Station-Id == 192.168.143.12`

En este ejemplo limitamos el acceso por Telnet al usuario ciscovty desde la IP 192.168.143.12.

Acceso por consola `ciscocon Auth-Type:=Local, Password == ``passcon'', NAS-Port-Type != Virtual`
Acceso por PPP `Router_B Auth-Type:=Local, Password == ``test'', Service-Type == Frame-User, Framed-Protocol==PPP`

Restringimos a que el uso del usuario Router_B con password test se haga desde una conexión PPP.

Acceso a modo privilegiado `$enab15$ Auth-Type:=System, Service-Type == Administrative-User, NAS-IP-Address == ``192.168.40.1''`

Ahora el usuario \$enab15\$ que Cisco utiliza para pasar a modo privilegiado (es equivalente al root en Unix) se autenticará usando el password del sistema, no mediante RADIUS. Esto lo hacemos con `Auth-Type:= System`. Ahora, el password se encuentra en la máquina que hace de servidor (generalmente encriptado en `/etc/password` o en `/etc/shadow` mediante DES y MD5 respectivamente), no en la base de datos del servidor RADIUS.

4.2 clients

El fichero clients almacena líneas que contienen el nombre del cliente y su secreto compartido:

```
NAS_1                SecretoNAS_1
NAS_2                SecretoNAS_2
192.168.23.41       SecretoNAS_41
```

El cliente se puede poner en forma de nombre de host o por su dirección IP. Se recomienda el uso de direcciones IP para evitar falseos de consultas DNS (DNS spoofing).

El secreto puede tener hasta 15 caracteres imprimibles ASCII (sin contar el espacio).

4.3 radius.conf

Define la configuración global de FreeRADIUS. Ver la página del manual para más información.

4.4 dictionary

Este fichero contiene una lista de los atributos y valores que el servidor RADIUS soporta. Permite añadirle una nueva característica al servidor cuando aparece una nueva funcionalidad en la parte cliente.

Ver la página del manual para más información.

5 Bibliografía

References

[1]cisco Systems, www.cisco.com

[2]FreeRADIUS website, www.freeradius.org

[3] ["RADIUS for UNIX Administrator's Guide"](#) , Lucent Technologies, Febrero de 1999.

[4] [RFC 2865](#) , Remote Authentication Dial In User Service (RADIUS),
[RFC 2866](#) , RADIUS Accounting,

[5] ["The Linux-PAM System Administrator's Guide"](#)
["The Linux-PAM Module Writer's Guide"](#)
["The Linux-PAM Application Developer's Guide"](#)