UNIVERSITÀ DEGLI STUDI DI CAMERINO

Facoltà di Scienze e Tecnologie

Corso di Laurea in Informatica

Dipartimento di Matematica ed Informatica

AAA in ambiente OpenSource: PPPoE, LDAP, RADIUS

Applicazioni - Implementazione - Risultati

Tesi di Laurea sperimentale

Laureando

Rocco De Marco

Relatore

Dott. Fausto Marcantoni

Anno Accademico 2004-2005

Indice

Prefazione

Quando qualcuno suona al campanello della nostra abitazione, attraverso il citofono normalmente rivolgiamo la classica domanda: "Chi è?". La nostra è formalmente una richiesta di identificazione per stabilire se autorizzare o meno il visitatore ad accedere presso il nostro domicilio.

Chi risponde può identificarsi più o meno correttamente, indicando, ad esempio, di essere l'addetto della compagnia del gas quando invece è il classico truffatore che non vede l'ora di mettere le mani sui nostri beni. Per cui è buona abitudine chiedere un documento o altre informazioni prima di accordare l'accesso.

Nel mondo dell'informatica una situazione analoga si ha nel controllo degli accessi a reti locali, a sistemi e servizi, banche dati e così via. Situazioni in cui l'accertamento dell'identità dell'utente riveste un importanza sempre maggiore, a patto che non sia una procedura complessa, macchinosa o addirittura onerosa per l'utilizzatore.

L'autenticazione degli utenti deve essere un giusto equilibrio tra sicurezza e flessibilità, tra scalabilità e economicità. Non si può chiedere loro di acconsentire ad un prelievo del sangue, da cui sarà analizzato il dna per accertarsi dell'identità, come non si può tollerare una rete configurata in modo tale che con un portatile munito di interfaccia wireless si acceda liberamente alla nostra rete senza alcuna autenticazione.

Un altro diffuso e inaccettabile problema è la proliferazione sterminata delle password. In alcune realtà un utente deve ricordarsi le varie password (e talvolta i relativi username) per usufruire di diversi servizi offerti dalla stessa organizzazione: dalla posta elettronica alla password di accesso alla postazione di lavoro, dalla password per la procedura X a quella per accedere

ad una sezione di un sito. Costringere un utente ad utilizzare decine di coppie di username e password significa arrivare ad una tipica reazione allergica: mettere ovunque la stessa password.

A questa strategia i system manager rispondono con una ulteriore "cattiveria": la scadenza delle password, che comporta per l'utente uno stress che tutto sommato non merita.

La soluzione spontanea sarebbe l'utilizzo della certificazione, magari dando a ciascun utente una bella smart card da inserire per utilizzare qualsiasi servizio. Affronteremo più avanti questo argomento, ma vanno anticipati sin d'ora alcuni aspetti che non favoriscono l'impiego dei certificati digitali in realtà eterogenee: il costo, la struttura necessaria, la scarsa flessibilità e il supporto non completo delle diverse categorie di utenza.

Allora cosa rimane da fare? La soluzione è semplice: date a Cesare una sola password. Ovvero, centralizzare i dati di identificazione su un unico sistema e utilizzarlo per tutti i servizi disponibili. A quel punto Cesare sarà ben contento di cambiare la password anche una volta al mese, e non dovrà più occupare la metà della rubrica del telefonino con username e password. La soluzione più flessibile per centralizzare informazioni, di autenticazione e non, degli utenti ad oggi risponde al nome di LDAP.

LDAP, che significa Lightweight Directory Access Protocol, è già ampiamente utilizzato per gestire l'autenticazione di numerosi servizi: posta elettronica, domini, servizi web, accessi a workstation, proxy server, e così via.

In queste pagine sarà analizzata una nuova frontiera: utilizzare LDAP per il controllo degli accessi nelle reti locali. Reti tradizionali, con il cavetto utp, o reti wireless. Appare chiaro che non è sufficiente l'impiego del solo ldap per raggiungere questo risultato, servono due attori molto importati: il PPP Over Ethernet (PPPoE) e un server RADIUS (Remote Access Dial In User System).

Si, si tratta dello stesso sistema impiegato per le connessioni ADSL e vedremo che è utilizzabile con risultati molto validi anche nelle reti locali. Risolvendo diverse problematiche di sicurezza, legate anche alle reti wireless, ma semplificando la vita degli utenti che con una semplice configurazione

potranno accedere in qualsiasi punto della rete senza problemi. Una connessione, questa, stabilita utilizzando quello username e quella password che vengono usati in tutti gli altri servizi offerti dall'organizzazione.

Quanto costa questo sistema? Dipende dalla soluzione implementata. Ci sono prodotti commerciali che costano anche 8000 dollari (granppprix), soluzioni hardware che prevedono l'impiego di router o dispositivi dedicati. Ma in queste pagine sarà illustrato come implementare questo sistema con software Open Source di libero utilizzo: rp-pppoe, freeradius e openldap.

Capitolo 1

FreeRADIUS

FreeRADIUS è una implementazione OpenSource del protocollo RADIUS (Remote Authentication Dial In User Service) che è a sua volta una implementazione del framework AAA (Autentication, Authorization, Accounting).

Il RADIUS server può essere visto come il piantone di una caserma che autorizza o meno l'accesso ad una risorsa, ma ne contabilizza anche alcuni parametri quali durata dell'erogazione del servizio, quantità di servizio erogato, e così via.

Entriamo in dettaglio analizzando nell'ordine il framework AAA, il protocollo RADIUS e quindi FreeRADIUS.

1.1 AAA

Il framework AAA ha come scopo, sostanzialmente, di fornire le risposte alle seguenti cruciali domande:

- "Chi sei?"
- "Dove sei?"
- "Quali servizi sono autorizzato ad offrirti?"
- "Per cosa hai utilizzato i miei servizi, come e per quando tempo?"

Queste sono le domande che il nostro AAA server pone a tutti i client che gli chiedono un servizio, valutando le risposte fornite e tenendo traccia delle loro attività.

AAA significa Autenticazione, Autorizzazione e Accounting.

1.1.1 Autenticazione

L'autenticazione è la richiesta che fa il server AAA al client di identificarsi, attraverso combinazione di username e passoword o attraverso l'impiego di certificati digitali. La conoscenza della passoword, o il possesso di un certificato riconosciuto valido dal server AAA, implicano l'accesso allo stadio successivo di autorizzazione.

Esistono numerosi modi per comunicare gli estremi di autenticazione. Tra questi figurano: pap, chap, peap, eap/tls, eap/ttls.

1.1.2 Autorizzazione

Una volta identificato l'utente i suoi dati vengono passati al modulo di autorizzazione, che deve stabilire cosa l'utente può fare o meno.

Ad esempio, nel caso di una connessione internet si può stabilire quale indirizzo IP utilizzare, qual è la durata massima di una sessione e il traffico massimo che può essere fatto. Un gestore di telefonia mobile GSM/UTMS potrebbe addirittura sfruttare un server AAA per non consentire ulteriori chiamate a chi ha esaurito il proprio credito.

Da un punto di vista prettamente di networking è possibile impostare regole di filtraggio di indirizzi ip, impostazione del routing, limitazioni di ampiezza della banda, utilizzo di comunicazioni con crittografia, e tanto altro ancora.

In alcune implementazioni di AAA, tra cui (free)RADIUS è possibile andare a prelevare i dati di autorizzazione direttamente dai database, rendendo infinite le possibilità di utilizzo di questo modulo.

Una volta che l'utente è autorizzato, la sua "pratica" passa in mano al modulo di accounting.

1.1.3 Accounting

Il modulo di accounting è il terzo ed ultimo del framework AAA, cui non spetta il compito di autorizzare o negare l'accesso a un utente o a un servizio: spetta il compito di tenere traccia delle sue attività.

Volendo, potremmo considerarlo come un ragioniere che tiene traccia della quantità di dati scambiati, la natura del servizio erogato, il tempo in cui un utente resta collegato al servizio, e il luogo da cui si è connesso.

Questi dati possono essere utilizzati per presentare all'utente la bolletta che sarà proporzionata alla quantità di servizio di cui ha usufruito.

L'accounting è una operazione che viene fatta in realtime, offrendo ai gestori dei servizi di sapere informazioni fondamentali, quali il numero di utenti connessi e valutare se ci sono anomalie nell'erogazione dei servizi.

Anche per il modulo di accounting è previsto in alcune implementazioni la possibilità di memorizzare le informazioni raccolte in un database.

1.1.4 Sequenze di autorizzazione

Il modello generico con cui abbiamo a che fare è costituito da tre entità: il client che effettua la richiesta, il server AAA che autorizza, il NAS[1] che rappresenta chi offre il servizio.

Esistono tre sequenze di autorizzazione, che variano sostanzialmente in funzione di chi tra AAA server e NAS deve rispondere al client.

La sequenza *agent* prevede che il client inoltri la sua richiesta al server AAA che a sua volta accede alla risorsa offerta dal NAS per poi restituire il responso al client. Una connessione con sequenza agent è illustrata in figura 1.1.

La sequenza *pull* prevede che il client comunichi direttamente con il NAS che a sua chiederà l'autenticazione dell'utente al server AAA (Figura 1.2).

[1] NAS: Network Access Server, da non confondersi con Network Attached Storage, che ha purtroppo la stessa sigla

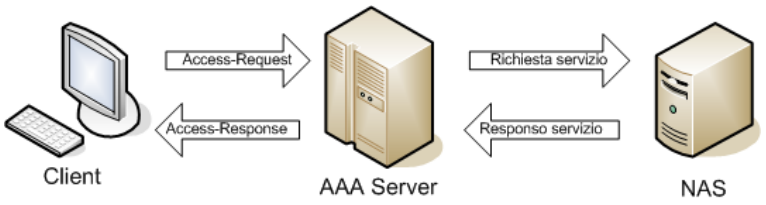


Figura 1.1: Sequenza Agent

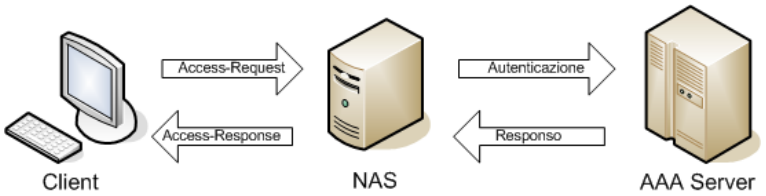


Figura 1.2: Sequenza Pull

La sequenza *push* prevede che il client si autentichi sul server AAA. Se l'autenticazione ha esito positivo gli viene restituito un token, che semplificando potremmo definire un lasciapassare, che sarà utilizzato in un secondo passaggio dal client per comunicare direttamente con il NAS (Figura 1.3).

La sequenza generalmente più usata è la pull, anche se ha lo svantaggio non irrilevante di far transitare le informazioni di autenticazione sul NAS. Un NAS mal configurato o violato potrebbe costituire una grossa falla in un sistema di autenticazione e le informazioni prelevate potrebbero essere utilizzate indebitamente da terzi.

Il metodo agent e il metodo push ovviano a questo difetto, ma di contro sono più difficili da implementare e richiedono oneri decisamente maggiori, soprattutto nella tipologia agent, per il server AAA.

Per l'autenticazione delle connessioni PPPoE è stato previsto l'utilizzo della sequenza di autenticazione pull.

1.1.5 Ulteriori aspetti del framework AAA

Il framework AAA prevede ulteriori features per aiutare gli sviluppatori nelle implementazioni. Vale la pena di accennare le funzioni di proxing.

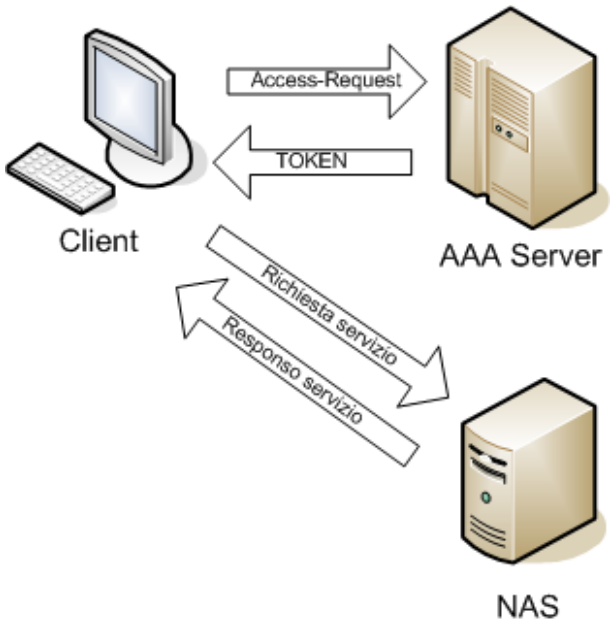


Figura 1.3: Sequenza Push

La funzione di proxing è molto importante quando le informazioni degli utenti sono partizionati su diversi server AAA. Può capitare che un azienda inglobi un altra, trovandosi due distinti gruppi di utenti sparsi in due distinti punti. Il problema è consentire agli utenti dell'azienda acquisita di sfruttare tutti i servizi disponibili nell'azienda principale.

Ci sono anche casi in cui sia conveniente, per motivi di prestazioni, partizionare gli utenti su diversi server AAA. Il caso più comune è quello degli ISP[2], che potrebbero dislocare territorialmente gli utenti.

Il proxing, che consente di risolvere questi problemi, è implementato in due modi diversi: in modo hop-to-hop e in modo end-to-end.

Il modello hop-to-hop prevede che ci siano delle relazioni di trust tra diversi server AAA. Qualora le credenziali fornite dal client non siano soddisfatte dal primo server AAA, esso le inoltra ad un secondo, che le inoltra ad un terzo e così via. Fino a che il client venga autenticato da un n-esimo server AAA o finchè non si arriva all'ultimo server AAA a disposizione. Se l'autenticazione avrà esito positivo su uno dei server AAA il responso verrà trasmesso a

[2]ISP: Internet Service Provider

ritroso (hop to hop) fino al primo server che comunicherà il responso al client.

Il modello end-to-end prevede che qualora il server AAA non sia in grado di autenticare un client, gli trasmetta gli estremi per connettersi ad un secondo server, continuando ricorsivamente in questo modo. Qualora uno dei server sia in grado di autenticare il client, sarà lui stesso a fornirgli il responso. Da qui il nome end to end.

1.2 RADIUS

RADIUS significa letteralmente Remote Authentication Dial In User Service. Il RADIUS è una implementazione del framework AAA, anche se il primo è venuto alla luce prima del secondo.

In effetti il framework AAA è stato disegnato sulla traccia del RADIUS, ma in modo da consentire più facilmente nuovi sviluppi.

Esistono evoluzioni di RADIUS, quali Diameter[3], che a differenza del RADIUS utilizza connessioni TCP, TACACS e TACACS+.

Stando alle specifiche RFC il protocollo RADIUS:

- Comunica con il protocollo UDP
- Usa il modello di sicurezza hop-to-hop
- Supporta l'autenticazione PAP e CHAP
- Usa MD5 per cifrare le password
- Suppota il modello di autenticazione, autorizzazione e accounting

Radius utilizza il protocollo UDP per comunicare, sulla porta 1812, con il client.

Il formato del pacchetto UDP è analogo a quello mostrato in figura 1.4. Analizziamo rapidamente il significato dei singoli campi:

[3]Si noti che il diametro è due volte il raggio

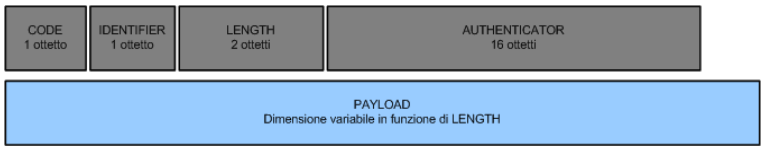


Figura 1.4: Formato del pacchetto radius

- Il campo *Code*, che contiene un ottetto, specifica il tipo di messaggio. Codici validi sono Access-Request (1), Access-Accept (2), Access-Reject (3), Accounting-Request (4), Accountin-Response (5) e Access-Challenge (11).
- Il campo *Identifier*, della dimensione di un ottetto, serve per distinguere i differenti client in caso di richieste simultanee.
- Il campo *Lenght* specifica la dimensione del messaggio. La dimensione può variare da 20 a 4096.
- Nel campo *Authenticator* sono contenute informazioni che consentono di controllare l'integrità del campo payload.
- Il campo *Payload* contiene le informazioni inviate dal client al server e vice-versa.

1.2.1 Tipologie di pacchetti

I pacchetti più importanti per le fasi di autenticazione AAA sono:

- Access-Request

  Contiene la richiesta del client di poter accedere ad un determinato servizio. Questo paccheto può contenere username, password, indirizzo ip, ecc.

- Access-Accept

  Contiene la risposta di autenticazione avvenuta con esito positivo, inviata dal server al client.

- Access-Reject

  Viene inviato dal server al client qualora la richiesta di autenticazione ha dato esito negativo.

- Access-Challenge

  Viene inviato dal server al client qualora qualcosa non quadra nel processo di autenticazione. Il pacchetto contiene le informazioni che il server vuole che siano nuovamente trasmesse tramite un nuovo pacchetto di Access.Request. In caso di mancata risposta da parte del client verrà inviato un pacchetto di Access-Reject.

1.3 FreeRADIUS

FreeRADIUS è una delle principali implementazione OpenSource di RADIUS server. Il software, giunto ormai alla versione 1.1.0, è rilasciato sotto licenza GNU/GPL, versione 2, ed è liberamente utilizzabile e prelevabile dall'indirizzo http://www.freeradius.org.

FreeRADIUS offre tutte le funzionalità offerte dai software commerciali. Attualmente freeRADIUS offre il supporto per LDAP, MYSQL, PostgreSQL, Oracle. Gestisce autenticazioni EAP (EAP-MD5, EAP-SIM, EAP-TSL, EAP-TTSL, EAP-PEAP), MSCHAP, MSCHAPV2, Cisco LEAP, oltre ai classici PAP e CHAP.

FreeRADIUS è basato sull'implementazione del Livingston RADIUS server, ma è stato completamente riscritto. Tra le features offerte sono disponibili:

- Le recenti versioni di freeradius sono fornite con un sitema di configurazione via web basato su pagine php.
- La possibilità di limitare il numero massimo di connessioni simultanee.
- Il supporto di più metodi di autenticazione in contemporanea.
- Permette la possibilità di vietare l'accesso a interi gruppi di utenti.
- Può eseguire programmi ad hoc ad accesso avvenuto.

- Il file di configurazione è modulare, si possono inserire le sezioni semplicemente con il comando *INCLUDE*.
- Supporta le specifiche dei più importanti produttori[4]. Tra cui: la serie Cisco Access Server, 3com/USR Netserver, Cistron PortSlave, Livingston PortMaster.
- Può operare come Proxy RADIUS server.

1.3.1 Installazione

La maggior parte delle distribuzioni linux offre freeRADIUS in pacchetti precompilati. Ad esempio su Debian GNU/Linux basta digitare il comando:

```
# apt-get install freeradius freeradius-dialupadmin
```

In ogni modo è possibile procedere all'installazione partendo dal file freeradius.x.y.z.tar.gz disponibile sul sito ufficiale. In tal caso l'installazione di base prevede alcuni semplici passaggi:

```
# tar xfvz freeradius.x.y.z.tar.gz
# cd freeradius.x.y.z
# ./configure
# make
# make install
```

1.3.2 Configurazione

L'ubicazione degli eseguibili, dei file di configurazione e di log varia da distribuzione a distribuzione. In linea di massima i files di configurazione dovrebbero trovarsi nel percorso /etc/raddb o /etc/freeradius.

Ecco l'elenco dei file installati su una distribuzione Debian GNU/Linux:

```
# ls /etc/freeradius
acct_users    experimental.conf  oraclesql.conf    realms
```

[4]La lista completa di produttori ne contiene più di cinquanta.

```
attrs          hints              otp.conf           snmp.conf
certs          huntgroups         otppasswd.sample   sql.conf
clients        ldap.attrmap       postgresql.conf    users
clients.conf   mssql.conf         preproxy_users
dictionary     naslist            proxy.conf
eap.conf       naspasswd          radiusd.conf
```

I files più importanti per la nostra configurazione sono:

```
radiusd.conf
clients.conf
sql.conf
```

Il file *radiusd.conf* contiene le direttive di configurazione del server RADIUS. Il file è diviso in diverse sezioni. La struttura principale è la seguente:

```
[opzioni globali]
modules{

}
authorize{

}
authenticate{

}
accountig{

}
```

Nella sezione modules vengono definiti i singoli moduli che possono essere indistintamente di autorizzazione, di autenticazione e di accounting.

Le sezioni *authorize*, *authenticate*, *accounting* contengono i nomi dei moduli precedentemente configurati. Se ad esempio vogliamo utilizzare il modulo pap per l'autenticazione, il file di configurazione deve contenere:

```
...
modules{
    pap {
         encryption_scheme = crypt
    }
}
...
authenticate{
    Auth-Type PAP {
         pap
    }
}
...
```

I moduli disponibili sono molti, basta dare un occhiata al file di configurazione fornito insieme alla distribuzione per rendersene conto. Tra i principali utilizzati nel nostro contesto sono:

```
modules{
    mschap{
    ...
    }
    ldap{
    ...
    }
    sql{
    ...
    }
}
```

Valuteremo in dettaglio questi moduli più avanti. È interessante osservare che il modulo sql può essere utilizzato in tutte le fasi: autenticazione, autorizzazione e accounting; infatti tutte queste informazioni possono essere lette da e scritte su un database.

Oltre le sezioni già illustrate, su radiusd.conf possono essere aggiunte anche le seguenti:

- listen{}
  dove possono essere inserite informazioni per specificare quale ip e quale porta utilizzare per il bind.
- security{}
  contiene impostazioni di sicurezza
- proxy server{}
  impostazioni per il proxing
- client IPADDRESS{}
  in cui vengono specificati i criteri per l'accesso da parte dei vari client
- tread pool{}
  stabilisce il numero di spare server ed il loro comportamento.
- instantiate{}
  sezione opzionale che contribuisce al caricamento dei moduli
- preacct{}
  sezione di pre-accounting: decide quale tipo di accounting deve utilizzare
- session{}
  per controllare gli utilizzi simultanei
- post-auth{}
  sezione che viene evalutata dopo che l'utente è stato autenticato
- pre-proxy{}
  contiene le impostazioni utilizzate per passare richieste autorizzative ad un altro server
- post-proxy{}
  ovvero cosa deve fare il server quando riceve un responso di proxing

1.3.3 Configurazione dei moduli mschap, ldap e sql

Per realizzare il sistema di autenticazione basato su PPPoE + freeRADIUS + openLDAP è necessario analizzare con maggiore dettaglio la configurazione dei moduli mschap, ldap e sql.

Come vedremo in seguito, utilizzeremo il protocollo MSCHAP per trasmettere i dati di autenticazione, openLDAP per contenere le informazioni degli utenti ed un database Mysql per archiviare le informazioni di accounting.

MSCHAP, e soprattutto MSCHAP-V2, è uns sistema sufficientemente sicuro per trasmettere le password, a differenza del PAP in cui vengono trasmesse in chiaro. In aggiunta viene sfruttato una connessione crittografica con protocollo MPPE[5] che rende ancora più sicura la transazione.

Ecco un esempio di configurazione di questo modulo:

```
mschap {
   authtype = MS-CHAP
   use_mppe = yes
   require_encryption = yes
   require_strong = yes
}
```

Il modulo è configurato per chiedere al client di instaurare una connessione con MPPE (use_mppe = yes) in modo obbligatorio (require_encryption=yes) e con chiave a 128 bit (require_strong=yes)

Bisogna ricordarsi di controllare se il Linux Kernel, sia del server che del client, sia compilato (o abbia il modulo) per gestire connessioni MPPE, altrimenti va installata una patch[6]. Recentemente le principali distribuzioni linux[7] danno la possibilità di impiegare kernel precompilati con il supporto MPPE.

Come già accennato il server RADIUS deve andare a prelevare le informazioni di autenticazione su una directory LDAP, che immaginiamo costituita dal dominio *testdoimain.it*. Un esempio di configurazione del modulo LDAP è il seguente:

```
ldap  {
       server = "localhost"
```

[5] MPPE: Microsoft Point-to-Point Encryption. Usa l'algoritmo di cifratura RC4

[6] La patch MPPE è disponibile all'indirizzo http://mppe-mppc.alphacron.de/

[7] Debian: 2.6.15-1, Suse: 2.6.13-15, Ubuntu: 2.6.12-9

```
        identity = "cn=admin,dc=testdomain,dc=it"
        password = testpass
        basedn = "ou=people,dc=testdomain,dc=it"
        filter = "(uid=%{Stripped-User-Name:-%{User-Name}})"
        start_tls = no
        dictionary_mapping = ${raddbdir}/ldap.attrmap
}
```

Analizziamo la configurazione del modulo:

- server
  specifica l'hostname o l'ip address del server LDAP. In questo caso il server LDAP è in esecuzione sulla stessa macchina

- identity
  contiene le informazioni di login per accedere sul server LDAP. Non sempre è indispensabile e dipende dalla configurazione del server LDAP stesso. In questo esempio si accede usando l'identificativo dell'amministratore, ma sarebbe meglio usare un utente apposito (magari chiamato radius) i cui permessi siano strettamente indispensabili per l'interrogazione.

- password
  contiene la password in chiaro dell'utente specificato. Per questa ragione bisogna configurare i permessi del file *radiusd.conf* in modo che sia solo l'utente radiusd (o analogo) a poter accedere in lettura su questo file.

- basedn
  contiene il riferimento alla struttura LDAP che contiene i dati degli utenti

- filter
  effettua la ricerca nella directory LDAP cercando tra i campi uid il valore ricevuto come User-Name.

- start_tsl
  la connessione con il server LDAP può essere fatta utilizzando la ci-

fratura TSL. In questo caso, trattandosi della stessa macchina, non è stata utilizzata.

- dictionary-mapping
  contiene il path e il nome de file che contiene il dizionario relativo a LDAP. Normalmente fornito nella distribuzione di freeRADIUS

L'autenticazione MSCHAP prevede che sia utilizzata non la password contenuta nell'attributo *userPassword* nella directory LDAP, ma la password *sambaNTPassword*. Vedremo che per utilizzare questo attributo è necessario aggiungere a openLDAP lo schema fornito con samba3. Ovviamente l'utente impostato in *identity* deve poter accedere a questo attributo.

Per poter sfruttare queste opzioni è necessario controllare che nel file ldap.attrmap siano presenti le seguenti linee:

```
checkItem       LM-Password       sambaLMPassword
checkItem       NT-Password       sambaNTPassword
```

Se siamo stati bravi nella configurazione a questo punto abbiamo tutti gli strumenti per autenticare gli utenti contenuti su una directory LDAP con freeRADIUS; ma vogliamo fare di più: inserire le informazioni di accounting su un database mysql[8].

Il modulo per configurare l'accesso ad un database potrebbe trovarsi in un file chiamato *sql.conf*, ma si può anche inserire il modulo direttamente nel file di configurazione principale. Questo modulo ha molte opzioni, che sono presenti già nel file di configurazione fornito con freeRADIUS. Le parti da modificare sono poche:

```
driver = "rlm_sql_mysql"
server = "localhost"
login = "mysql"
password = "mysqlpass"
radius_db = "radius"
```

Analizziamone il significato:

[8] Per la configurazione delle tabelle mysql si rimanda alla pagina 112

- driver
  Specifica il driver da utilizzare in base al DBMS impiegato. Opzioni valide sono: rlm_sql_mysql, rlm_sql_postgresql, rlm_sql_iodbc, rlm_sql_oracle, rlm_sql_unixodbc, rlm_sql_freetds. In questo caso abbiamo deciso di impiegare mysql.

- server
  Contiene l'hostname o l'ip address del server mysql. In questo caso è in esecuzione sulla stessa macchina.

- login
  Username usato per la connessione con il DBMS.

- password
  Contiene la passoword di accesso al DBMS in plain text.

- radius_db
  indica il nome del database che contiene la tabelle usate da freeRADIUS.

Per configurare il server freeRADIUS in modo da autenticare con MSCHAP e LDAP e per loggare l'accountig su mysql il file *radiusd.conf* sarà così strutturato:

```
...
modules {
   mschap {
      authtype = MS-CHAP
      use_mppe = yes
      require_encryption = yes
      require_strong = yes
   }
   ldap {
      server = "localhost"
      identity = "cn=admin,dc=testdomain,dc=it"
      password = testpass
      basedn = "ou=people,dc=testdomain,dc=it"
```

```
      filter = "(uid=%{Stripped-User-Name:-%{User-Name}})"
      start_tls = no
      dictionary_mapping = ${raddbdir}/ldap.attrmap
   }
   $INCLUDE ${confdir}/sql.conf
}
authorize {
   ldap
   mschap
}
authenticate{
   mschap
}
accounting {
   sql
}
```

Con questo chiudiamo questa breve panoramica sulle opzioni di configurazioni del file *radiusd.conf*. Maggiori informazioni sono reperibili nelle documentazione fornita con freeRADIUS e sul sito ufficiale.

1.3.4 Il file clients.conf

Il file *clients.conf* contiene le informazioni inerenti i client. Si tratta, in effetti, di sezioni di configurazione di *radiusd.conf* e possono anche esservi inserite direttamente.

In passato queste informazioni potevano essere inserite nel file clients, ma l'utilizzo odierno è deprecato.

lo schema di configurazione è il seguente:

```
client 127.0.0.1 {
     secret = testpass
     shortname = localhost
     nastype = other
}
```

Tutti i client che accedono al nostro RAADIUS server devono essere elecati in questo file. Al posto dell'indirizzo ip può anche essere utilizzato l'hostname del client.

I parametri di configurazione sono i seguenti:

- secret
  contiene la password in plaintext che i client dovranno utilizzare per connettersi al server radius. La lunghezza massima è di 31 caratteri.
- shortname
  viene utilizzato come alias da sostituire all'hostname o all'indirizzo ip. Questo parametro è obbligatorio
- nastype
  specifica il metodo per comunicare con il NAS. Opzioni valide sono: cisco, computone, livingston, max40xx, multitech, netserver, pathras, patton, portslave, tc, usrhiper, other.

1.3.5 Il tool dialupadmin

Le recenti versioni di freeRADIUS sono fornite con un interessante tool di gestione web-based chiamato *dialupadmin*. Con questa interfaccia web è semplice ed intuitivo consultare informazioni di accounting (che devono essere inserite su un database), informazioni sugli utenti connessi, traffico generato, stato del sistema, log degli errori e tanto altro. Tuttavia per la gestione degli utenti preferisco utilizzare *phpmyadmin* di cui discuteremo in seguito.

In figura 1.5 si possono vedere delle statistiche di accesso al nostro server RADIUS.

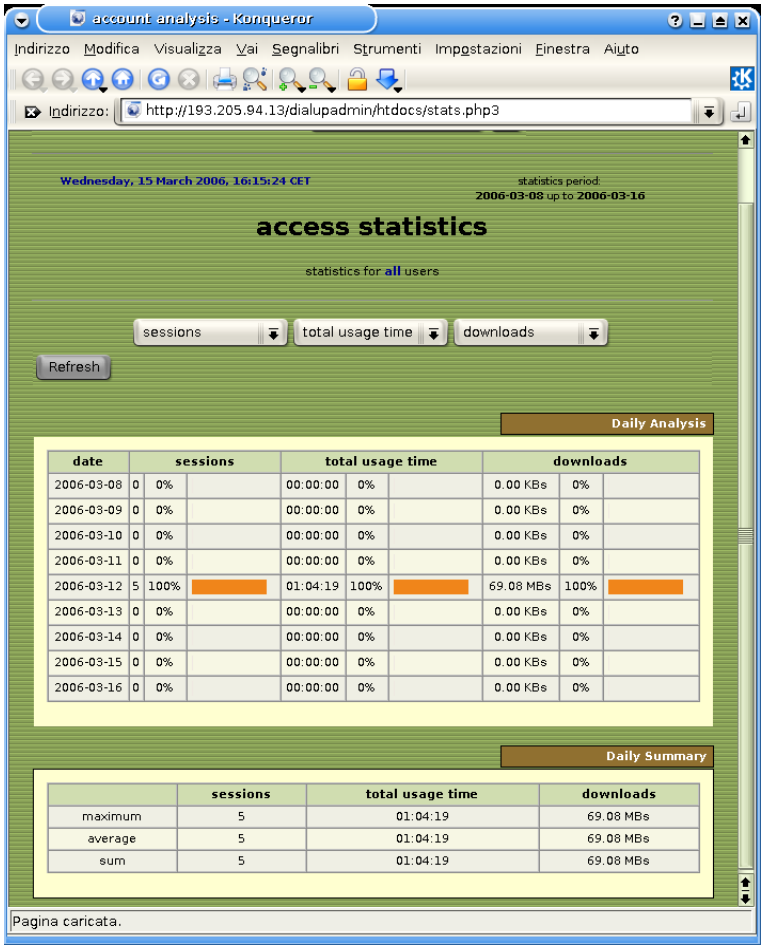


Figura 1.5: dialupadmin: statistiche di accesso

Capitolo 2

PPP Over Ethernet

PPP over Ethernet, di seguito chiamato PPPoE, offre la possibilità di connettere una rete di computer su un concentratore di accessi. Con questo modello ciascun host utilizza il suo personale stack PPP.

PPPoE è un protocollo per connessioni punto-punto, che consente di incapsulare frame PPP all'interno di frame Ethernet. Il protocollo opera a livello data link e consente di incapsulare a sua volta tutti i protocolli del livello network, come ad esempio IP, IPX, AppleTalk, su una linea seriale asincrona.

Il protocollo PPPoE non è uno standard Internet, ma è comunque illustrato nella RFC 2516 (A Method for Transmitting PPP Over Ethernet - PPPoE).

Il PPPoE è molto diffuso per gestire le connessioni DSL da parte degli ISP, ma vedremo che può essere impiegato con numerosi vantaggi anche all'interno di partizioni di reti locali. Come il PPP offre caratteristiche comuni di autenticazione, compressione e crittografia che possono essere utilizzate proprio per gestire e proteggere le connessioni dei client.

Il problema principale del PPPoE è la dimensione massima dell MTU[1] che è inferiore a quello dello standar ethernet. Questo causa frammentazione dei pacchetti con riduzione delle performaces di comunicazione.

[1] MTU: Unità massima di trasmissione, specifica le dimensioni massime di un pacchetto che può essere trasmesso.

Gli aspetti più interessanti che il protocollo PPPoE eredita dal PPP sono la possibilità di ottenere informazioni sul traffico generato dagli utenti, e il poter offrire numerose possibilità di configurazione: limitare l'ampiezza di banda, assegnare indirizzi ip in modo statico, e così via.

In questo capitolo sarà analizzato in dettaglio il protocollo e si introdurrà l'implementazione Open Source rp-pppoe.

2.1 Il protocollo PPP

2.1.1 Basi di PPP

Il ppp (protocollo punto-punto) è uno schema di incapsulazione che consente di trasmettere datagrammi attraverso linee seriali. A seconda del tipo di collegamento può essere di tipo sincrono o asincrono.

Il frame PPP contiene una intestazione ed una coda, e può trasportare, come già accennato, pacchetti IP e di altri protocolli. In figura 2.1 è mostrato un frame PPP che incapsula un pacchetto IP.

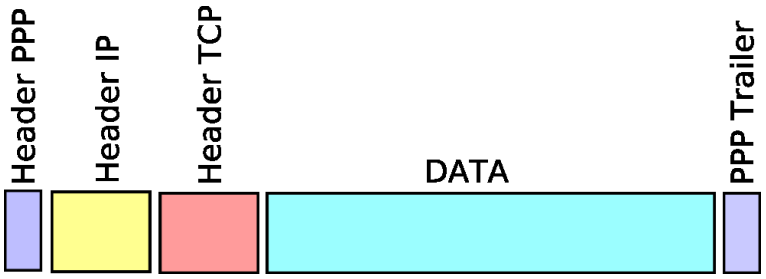


Figura 2.1: Pacchetto TCP/IP incapsulato in un frame PPP

Il campo trailer è molto importante, poichè rende possibile un rilevamento degli errori di trasmissione.

In figura 2.2 è mostrato in modo dettagliato un frame PPP. Analizziamo i vari campi:

- FLAG
  Campo della dimensione di un byte, è posto in testa e in coda al pacchetto è impostato con valore 0x7e (01111110).

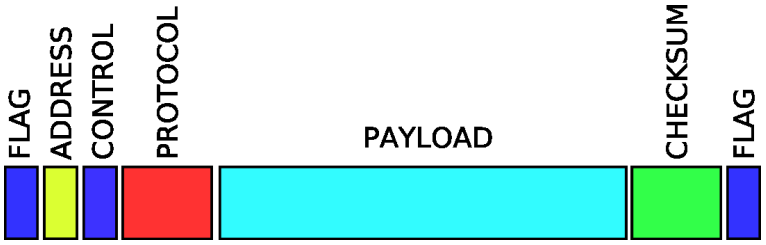


Figura 2.2: Formato del frame PPP

- ADDRESS
  Lungo un byte, contiene un indirizzo di broadcast (0xff) per consentire a tutte le stazioni di ascoltare.
- CONTROL
  Byte di controllo, il cui valore base è 0x03.
- PROTOCOL
  Indica il protocollo contenuto nel campo data. Può occupare 1 o 2 byte.
- PAYLOAD
  Campo dati, della lunghezza base di 1500 byte.
- CHECKSUM
  Lungo da 2 a 4 byte consente di verificare la correttezza del pacchetto inviato. Viene anche chiamato CRC o FSC (Frame Check Sequence).

I campi ADDRESS e CONTROL, che sono sempre costanti nella configurazione di base, possono anche non essere trasmessi (con alcuni accorgimenti del LCP) risparmiando 2 byte.

PPP offre un protocollo di controllo del collegamento, chiamato LCP (link controlo protocol) che consente di testare la linea, negoziare e controllare le opzioni di trasmissione. I tipi di pacchetto LCP sono illustrati in tabella 2.1.

Un altro protocollo offerto da PPP è il Network Control Protocol (NCP) che consente di negoziare opzioni del livello rete quali, ad esempio, l'assegnamento e l'amministrazione degli indirizzi IP. NCP è definito in base al

Configure-request	C → S	Elenco dele opzioni di configurazione proposte
Configure-ack	C ← S	Opzioni di configurazione accettate
Configure-nak	C ← S	Opzioni non accettate
Configure-reject	C ← S	Opzioni non disponibili
Terminate-request	C → S	Richiesta di disconnessione
Terminate-ack	C ← S	Disconnessione effettuata
Code-reject	C ← S	Richiesta non supportata
Protocol-reject	C ← S	Protocollo non supportato
Echo-request	C → S	Rispedire il frame
Echo-reply	C ← S	Frame rispedito
Discard-request	C → S	Frame da scartare

Tabella 2.1: Tipo di pacchetti LCP

protocollo di rete che deve asservire. Nel caso del protocollo IP assume il nome di IPCP.

2.1.2 PPP: operazioni di collegamento

Le fasi di una connessione PPP sono tipicamente quattro:

1. Connessione (Link Establishment Phase);
2. Autenticazione (Authentication Phase);
3. Configurazione Protocollo di rete (Network-Layer Protocol Phase);
4. Terminazione della connessione (Link Termination Phase).

Lo schema di collegamento prevede che il client parta dallo stato *Dead*, se riesca a stabilire la connessione passa all'*autenticazione*, altrimenti torna allo stato *Dead*. Se supera l'*autenticazione* passa allo stato *Network*, altrimenti la connessione viene terminata. Questo schema è illustrato in figura 2.3

Il protocollo di controllo del collegamento (LCP) è usato per stabilire il collegamento attraverso uno scambio di pacchetti, come descritti nella tabella 2.1. LCP non si occupa di configurare opzioni di network, cosa che avverrà a seguito dell'autenticazione.

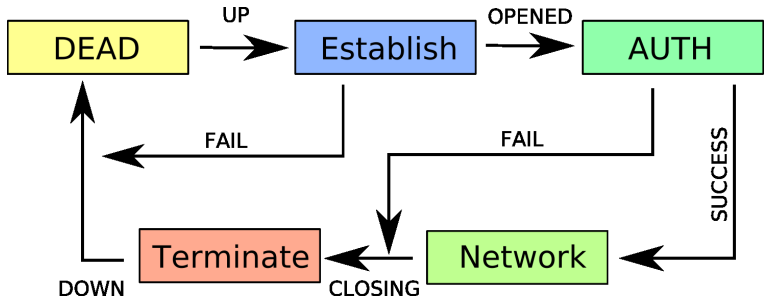


Figura 2.3: Schema di connessione PPP

L'autenticazione dovrebbe essere fatta non appena il collegamento è instaurato ed è possibile impostare un timeout oltre il quale la connessione viene chiusa. L'avanzamento dalla fase di autenticazione a quella di network non deve avvenire finchè l'autenticazione non è riuscita. Finchè ciò non accade tutti i pacchetti di protocolli diversi dall'LCP dovrebbero essere scartati.

La fase di configurazione del livello network è compiuto dall'appropriato NCP. La configurazione compiuta da NCP è relativa al protocollo impiegato. Nel caso di protocollo IP, NCP si occuperò di configurare parametri quali indirizzo IP, server DNS, gateway, netmask, etc.

Quando la connessione va terminata, o per esplicita richiesta da parte del client, o per decisione del server ppp in base a politiche prestabilite, si passa allo stadio di terminazione della connessione, a cui segue la disconnessione del client.

2.2 Il protocollo PPPoE

Come già accennato il frame PPPoE, mostrato nella figura 2.4, viene incapsulato in un frame ethernet. Si comporta alla stregua del protocollo PPP, offrendo le caratteristiche di autenticazione, cifratura e compressione dei dati.

In pratica il PPPoE è un protocollo di tunnelling tra due host, in cui il flusso di informazioni viaggia su una connessione ethernet e viene incapsulato con il protocollo PPP.

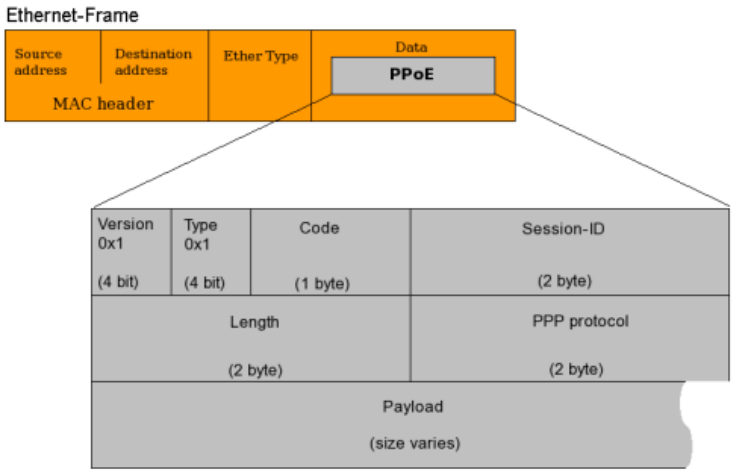


Figura 2.4: Frame PPPoE

Per le caratteristiche ereditate dal protocollo PPP, ovvero la possibilità di contabilizzare informazioni sul traffico ed effettuare un controllo sull'integrità dei dati trasmessi, è ampiamente usato dagli ISP per gestire le connessioni xDSL. In realtà questo protocollo può essere utilizzato indiscriminatamente quando tra due host è presente una connessione ethernet, come ad esempio una rete locale.

I vantaggi sono legati alla possibilità di gestire in modo razionale le connessioni degli utenti, differenziandone eventuali categorie, e proteggendo le comunicazioni con dei sistemi di cifratura. Torneremo in seguito su questo argomento per analizzarlo meglio.

Il protocollo PPPoE è composto da due distinti stadi: *Discovery* e *Session*. Il primo consente di instaurare la connessione, il secondo di gestirla.

2.2.1 Struttura del frame PPPoE

La struttura del frame PPPoE è mostrato in figura 2.4. Analizziamo il contenuto dei singoli campi:

- VER
  In base alle specifiche[2] PPPoE deve essere impostato a 0x01.

[2]RFC2516

- TYPE
  Anche questo campo deve essere impostato a 0x01 in base alle specifiche.
- CODE
  Campo utilizzato per specificare le fasi degli stadi di discovery e di sessione.
- SESSION_ID
  Viene impostato con il discovery, e vine conservato per tutta la durata della sessione, in unione alle informazioni SOURCE_ADDR, DESTINATION_ADDR dell'intestazione ethernet.
- LENGTH
  Specifica la dimensione del campo PAYLOAD.
- PPP PROTOCOL
  Contiene informazioni sul protocollo PPP, usato solo durante lo stadio di sessione.
- PAYLOAD
  Campo contentente i dati da trasmettere.

2.2.2 PPPoE: il Discovery

Quando un client vuole connettersi, effettua un discovery per ottenere il MAC address del concentratore di accessi con cui connettersi. Durante tutte le fasi di discovery il campo ETHER_TYPE dell'intestazione ethernet è impostato a 0x8863.

Le fasi di *discovery* sono ben definite, come mostrato in tabella 2.2

FASE 1	C → S	PADI: PPPoE Active Discovery Initiation
FASE 2	C ← S	PADO: PPPoE Active Discovery Offer
FASE 3	C → S	PADR: PPPoE Active Discovery Request
FASE 4	C ← S	PADS: PPPoE Active Discovery Session-confermation PADT: PPPoE Active Discovery Terminated

Tabella 2.2: Le fasi del discovery PPPoE

1. Il client invia dei messaggi (PADI) in broadcast sulla ethernet alla ricerca di un concentratore di accessi.

2. Il concentratore di accessi invia un messaggio (PADO) per informare il client della sua esistenza.

3. Il client effettua una richiesta (PADR) al concentratore di accessi per instaurare una sessione.

4. Se il Concentratore di accessi concede una sessione al client invia un messaggio di conferma (PADS), altrimenti invia un messaggio di terminazione (PADT). Il frame di terminazione (PADT) può comunque essere inviato in qualsiasi momento anche per interrompere una sessione.

2.2.3 PPPoE: Sessione

Quando sono noti gli indirizzi MAC di entrambi i pears (server e client) ed è stato impostato un numero di sessione, è possibile instaurare una sessione PPP, i cui frame sono ovviamente incapsulati con il PPPoE. Durante una sessione il campo ETHER_TYPE di ciasun frame ethernet è impostato con il valore 0x8864.

Da questo punto in poi la comunicazione viene gestita come una normale comunicazione PPP, con alcune differenze:

- I pacchetti sono trasmessi su Ethernet, invece che su linee seriali.
- Viene aggiunto un overhead di 6 byte sull'header ethernet.
- Non è utilizzato il checksum del PPP in quanto ethernet dispone di un suo proprio CRC.

L'impiego del PPPoE introduce importanti vantaggi:

- Introduce il concetto di sessione in una connessione ethernet. Questo può essere sfruttato per pianificare l'uso delle risorse in base a politiche personalizzabili a livello di utenti o gruppi.

- Una sessione PPP può richiedere l'autenticazione, cosa che consente di restringere l'accesso alle proprie risorse ad utenti ben identificati, vietando l'accesso a tutti gli altri.
- PPPoE può incapsulare anche protocolli non IP.

Nel prossimo capitolo analizzeremo una implementazione Open Source del PPPoE disponibile in ambiente Linux: rp-pppoe.

Capitolo 3

Linux e PPPoE

Se nel mondo Microsoft una delle più diffuse soluzioni per gestire connessioni PPP (e PPPoE) è il servizio Remote Access Service (RAS), in ambiente linux il software più quotato, ma anche più sviluppato, è Roaring Penguin PPPoE (rp-pppoe), distribuito liberamente con licenza Open Source.

Gli sviluppatori di questo software si sono trovati davanti ad una scelta sostaziale: se sviluppare gli stadi di discovery e session del protocollo PPPoE in *user-space* o in *kernel-space*. La loro scelta è stata quella di includere nel kernel entrambi, ma anche di sviluppare dei programmi da eseguire in *user-space* per evitare di ricompilare il kernel per abilitare il supporto PPPoE. Questo ha consentito anche il porting di rp-pppoe su piattaforme UNIX like, quale ad esempio freeBSD.

Attualmente i kernel linux forniti con le principali distribuzioni offrono un modulo PPPoE precompilato e pronto all'uso. Tuttavia ci sono diverse filosofie d'utilizzo di rp-pppoe: qualcuno lo preferisce eseguiro in *user-space*, altri in *kernel-space*.

Il sistema probabilmente migliore è di sfruttare il fatto che gli stadi sessione e discovery siano stati implementati indipendentemente ed in entrambi i modi. In questo modo si può eseguire le fasi di discovery, che sono meno onerose in termini di prestazioni, in user-space e la gestione delle sessioni affidata al kernel, che dovrebbe garantire performances più alte.

3.1 Funzionamento di rp-pppoe

Il software rp-pppoe si frappone tra l'interfaccia ethernet e il demone pppd, che creerà e gestirà una interfaccia ppp$n$. L'implementazione di PPP che può essere usata con il linux è quella disponibile sul sito http://ppp.samba.org, arrivata oggi alla versione stabile 2.4.3, anche se lo sviluppo appare fermo da diverso tempo.

Il demone pppd normalmente crea un network device (ppp$n$) e lo collega a un dispositivo seriale (tty). I pacchetti inviati al dispositivo ppp$n$ sono trasmessi sul dispositivo tty, e viceversa. Nel caso del PPPoE vengono impiegati degli *pseudo-tty* al posto dei dispositivi seriali.

Il demone pppd e il kernel si aspettano di comunicare con un dispositivo tty, ma vengono invece interfacciati con questo modulo pseudo-tty che comunica, attraverso standard input e standard output, con il demone pppoe, che a sua volta, sfruttando il *Raw Ethernet Socket*, è in collegamento con l'interfaccia ethernet eth$x$. Si faccia riferimento alla figura 3.1

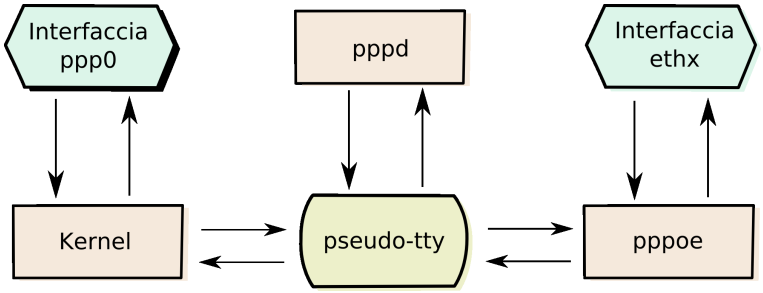


Figura 3.1: Funzionamento di rp-pppoe

3.1.1 Installazione e configurazione di rp-pppoe

A causa dell'elevata diffusione delle connessioni xDSL, a cui ci si collega prevalentemente con PPPoE, le principali distribuzioni linux sono già munite del software rp-pppoe di default nelle installazioni standard. In ogni modo è necessario utilizzare una versione di pppd ed un kernel che supportino il PPPoE. Per maggiori informazioni si rimanda alla documentazione

disponibile per ciascuna distribuzione linux, ma vediamo come provvedera all'installazione su un Debian GNU/Linux:

```
# apt-get install pppoe
```

Tipicamente tra i programmi installati troviamo: *pppoe*, *pppoe-server*, *pppoe-sniff*, *pppoe-relay*, oltre alla documentazione, le pagine del manuale e dei file di configurazione preimpostati. Il programma *pppoe* è un PPPoE client eseguito nello user-space, *pppoe-server* è il servente user-space, *pppoe-sniff* è un packet sniffer che consente di ottenere informazioni utili al debug.

La configurazione di pppoe-server è contenuta, normalmente, nel file /etc/ppp /pppoe-server-options. Per la configurazione è necessario stabilire a priori delle impostazioni che si vorrano poi utilizzare nella connessione ppp. Principalmente va stabilito quale protocollo di autenticazione utilizzare, se utilizzare compressione e/o cifratura

Il file pppoe-server-option potrebbe essere configurato come segue:

```
mtu 1490
mru 1490
auth
require-pap
```

Mentre il file di configurazione di pppd, /etc/ppp/options, può essere configurato con:

```
lock
ipcp-accept-local
ipcp-accept-remote
local
noreplace-defaultroute
plugin radius.so
```

In questo contesto non si vuole illustrare singolarmente tutte le opzioni di configurazione, per le quali si rimanda alla consulatazione delle pagine di manuale di pppd e pppoe, ma vale la pena di soffermarsi su alcune di esse.

I valori di *mtu* e *mru*[1] sono impostati a 1490 a causa delle caratteristiche del protocollo PPPoE, che non consente di sfruttare interamente il campo dati del frame ethernet, a causa dell'overhead. Questo, come è ben noto, causa problemi di frammentazione.

Il parametro *auth* specifica che deve essere richiesta una autenticazione per ottenere l'accesso. I dati di autenticazione dovranno essere inviati attraverso il protocolo *pap* in quanto così stabilito dalla direttiva *require-pap*. Vale la pena aprire una panoramica sui vari metodi di autenticazione:

- PAP (Password Autentication Protocol)
  Prevede la verifica delle informazioni di accesso attraverso uno scambio in chiaro della password. Assolutamente insicuro e da evitare in ogni caso. Basta un semplice sniffer per riuscire a catturare il pacchetto contenente le informazioni di autenticazione, contenute i uno dei frame scambiati tra client e server

```
0000  00 30 1b 3a 0b 6e 00 11  2f c4 cd 9c 88 64 11 00   .0.:.n.. /....d..
0010  00 0b 00 1f c0 23 01 0a  00 1d 0d 72 6f 63 63 6f   .....#.. ...rocco
0020  2e 64 65 6d 61 72 63 6f  0a 70 69 70 70 6f 62 61   .demarco .pippoba
0030  75 64 6f ff ff ff ff ff  ff ff ff ff              udo..... ....
```

- SPAP (Shiva PAP)
  Utilizza un algoritmo di cifratura della password che è però reversibile. Il sistema di protezione è comunque misero e va comunque scartato.

- CHAP (Challenge Handshake Authentication Protocol)
  Prevede la trasmissione da parte del server di un *hash*, calcolata dalla password dell'utente attraverso l'algoritmo MD5, e quindi verificata dal client con la password inserita dall'utente. Il sistema è più sicuro dei precedenti, perlomeno riguardo lo scambio delle passowrd, ma, come vedremo in seguito, non consente di instaurare connessioni cifrate.

- MS-CHAP (Microsoft CHAP)
  Utilizza un meccanismo simile al CHAP, ma impiega MD4 per l'hashing della password e una cifratura DES tra client e server.

[1] MRU: Maximum Receive Unit

- MS-CHAP-v2
  Simile al MS-CHAP, ma utilizza l'algoritmo di cifratura 3DES per lo scambio delle password. In congiunzione con MPPE (Microsoft Point-to-Point Encription) consente di stabilire delle connessioni relativamente sicure.

L'ultima riga del file di configurazione /etc/ppp/option data in esempio, abilita la comunicazione con un server RADIUS per provvedere all'autenticazione. Questa operazione è compiuta dal radiusclient, di cui parleremo in seguito.

3.1.2 Utilizzo di pppoe-server

Il programma *pppoe-server* ci consente di avviare il demone che gestisce le connessioni PPPoE. Le opzioni principali che possono essere passate come parametri sono:

-I *interface* specifica su quale interfaccia ethernet ricevere le connessioni.

-C *nome* specifica quale deve essere assegnato al server come concentratore di accessi.

-S *nome* specifica il nome del servizio offerto.

-L *IP* indirizzo IP locale da assegnare all'interfaccia virtuale ppp*n* del server.

-R *IP* indirizzo IP base del range di quelli disponibili ad essere assegnati ai client.

-N *num* specifica il numero massimo di accessi concorrenti. Di default è 64.

-T *num* specifica il numero di secondi di timeout.

-r Chiede a pppoe-server di permutare randomicamente i session numbers.

-k Avvia il pppoe-server in kernel mode.

A titolo di esempio il pppoe-server può essere avviato con le seguenti opzioni:

```
# /usr/sbin/pppoe-server -k -I eth0 -N 50 -C myNAS -R 10.20.30.1
```

Questo comando farà avviare pppoe-server in kernel-mode e sarà disponibile a 50 client a cui saranno assegnati indirizzi ip compresi nel range 10.20.30.1 - 10.20.30.51. Il binding avverrà sulla interfaccia ethernet eth0.

3.2 Configurazione dei client

La configurazione dei client che desiderano utilizzare PPPoE è una procedura semplice e che avviene per mezzo di tool molto intuitivi, sia in ambiente Windows, che Mac OS, che Linux (e gli altri unix like). Questo perchè, come detto, ha contribuito la sempre maggiore diffusione delle connessioni xDSL.

3.2.1 Client linux

Su linux è disponibile un comando chiamato *pppoeconf* che, dopo aver fatto un discovery, chiederà all'utente tutte le informazioni, principalmente di autenticazione, necessarie per collegarsi. Su alcune distribuzioni linux esistono delle apposite GUI che vanno ad interfacciarsi con questo software, ma il risultato è analogo.

Una volta configurata la connessione viene creato un file, tipicamente chiamato *dsl-provider* e collocato nel percorso /etc/ppp/pears, che contiene tutte le informazioni di accesso. Lo username e la password per il collegamento saranno salvate nel file pap-secret e/o nel file chap-secret, entrambi posti nella directory /etc/ppp. Visto che la password viene salvata in chiaro all'interno di questi file è necessario settare gli opportuni permessi per evitare spiacevoli problemi.

Per gestire la connessione esistono tre semplici comandi:

pon ***nomeconnessione*** Abilita la connessione specificata. Di solito è chiamata *dsl-provider*.

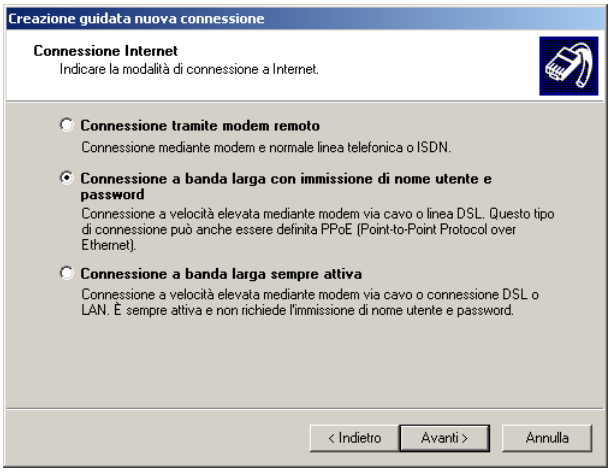
poff ***nomeconnessione*** Disabilita la connessione.

plog Offre informazioni di log sulla connessione.

3.2.2 Client windows

Le recenti versioni di Microsoft Windows forniscono nativamente il supporto PPPoE. La configurazione è molto semplice, e può essere fatta utilizzando lo strumento *Crea una nuova connessione* nella finestra delle connessioni di rete.

In linea di massima è sufficiente inserire solamente lo username e la password per potersi collegare al server PPPoE.



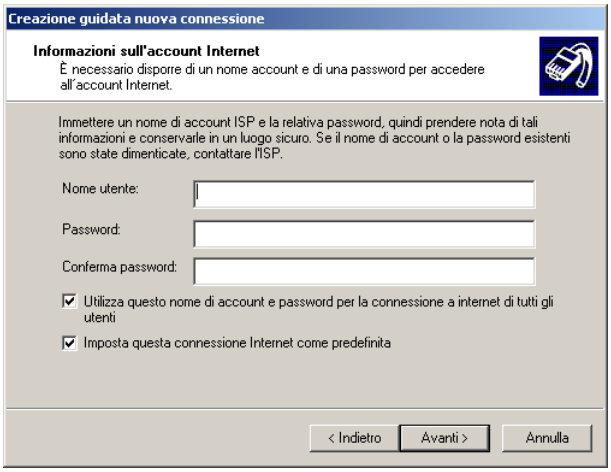


Figura 3.2: Configurazione del client PPPoE su Windows Xp

3.3 Connessione PPPoE

Una volta che client e server sono ben configurati è possibile instaurare una connessione PPPoE. Nel nostro caso abbiamo l'intenzione di utilizzare degli utenti i cui dati di autenticazione sono inseriti su una directory LDAP a cui accederemo, come già illustrato nel Capitolo 1, utilizzando il server freeRADIUS.

3.3.1 MSCHAPv2 e MPPE

Per motivi di sicurezza la scelta del protocollo di autenticazione è caduta su MSCHAPv2, ma occorre verificare che il demone pppd sia abilitato a gestirlo. Per utilizzarlo bisogna inserire le seguenti righe nel file di configurazione pppoe-server-option:

```
refuse-pap
refuse-chap
refuse-mschap
require-mschap-v2
```

Un altro problema da risolvere è quello della sicurezza della comunicazione. Normalmente tutte le informazioni viaggiano in chiaro, e, anche se non è facile carpire le informazioni di autenticazione, tutti gli altri dati che vengono scambiati (email, pagine web, etc.) potrebbero essere intercettati. Soprattutto se si tratta di una connessione wireless.

Per arginare il problema possiamo usare il protocollo di cifratura MPPE (Microsoft Point-to-Point Encryption), di cui sono tuttavia noti alcuni possibili problemi di sicurezza; ma che offre una soluzione comunque accettabile di sicurezza.

Si ricorda che per poter utilizzare MPPE è necessario disporre di un kernel che lo supporti. Poi è possibile inserire le seguenti informazioni nel file di configurazione pppoe-server-option:

```
require-mppe
```

Il protocollo MPPE supporta una cifratura a 40bit e una più strong a 128bit. Esistono altri parametri di configurazione per forzare l'adozione di una o dell'altra.

3.4 Il radiusclient

Per autenticare gli utenti che accedono al nostro NAS utilizzando un RADIUS server è necessario utilizzare un client RADIUS, che è disponibile in pacchetti precompilati per le principali distribuzioni linux. Per installarlo, ad esempio, su Debian basta digitare il comando:

```
# apt-get install radiusclient1
```

Per utilizzarlo con successo è necessario configurare opportunamente alcuni file, di cui i più importanti sono *radiusclient.conf* e *servers*. Entrabi questi files dovrebbero trovarsi al percorso /etc/radiusclient.

Il file *radiusclient.conf* contiente parametri di configurazione, mentre il file *servers* l'elenco dei server su cui tentare l'autenticazione.

Ci sono delle avvertenze di cui tenere conto se si vuole utilizzare radiusclient per autenticare con il protocollo MSCHAPV2, come illustrato nell'appendice A alla pagina 109.

Capitolo 4

LDAP e OpenLDAP

LDAP, Lightweight Directory Access Protocol, è sostanzialmente un protocollo di accesso ad una stuttura dati ad albero, da cui il nome di directory. LDAP nasce come metodo di accesso semplificato (leggero) ai servizi di directory X500, standardizzate dall'ISO, rispetto al protocollo originario DAP (Directory Access Protocol).

La stuttura di una directory LDAP è gerarchica e piramidale. Ciascun nodo, chiamato *entry* ha una collezione di attributi ed è individuato univocamente e referenziato con gli altri. I diversi tipi di *entry* avranno una collezione di attributi diversa in base a ciò che astrattizzano.

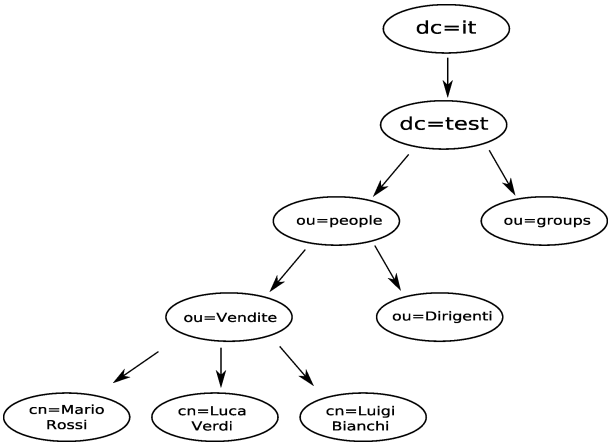


Figura 4.1: Struttura di una directory LDAP

Prendiamo ad esempio la struttura LDAP mostrata in figura 4.1, dove è rappresentata una ipotetica azienda italiana chiamata *test* in cui le persone che vi lavorano sono divise in due categorie, *Vendite* e *Dirigenti*.

4.1 Le directory X.500

Probabilmente l'esigenza di avere degli elenchi stutturati, efficienti e rapidi da consultare, è nata con la diffusione della telefonia. Come si può fare a telefonare ad un parente, di cui si conosce il nome ma non il numero di telefono, che abita in un paese diverso dal nostro? La soluzione è semplice: si prende l'elenco telefonico, ammesso di averlo a disposizione, della provincia in questione, si cerca la pagina relativa al paese del parente e si cerca il suo nominativo tra la lista di persone, che sono ordinate alfabeticamente.

Formalizzando il modello, l'elenco telefonico è una directory organizzata per *provincia*, *comune* e *nomi di persona*. Ciascuna *entry* relativa ad una persona contiene tipicamente tre attubuti: *nominativo*, *indirizzo* e *numero di telefono*. La ricerca è relativamente rapida e semplice: già un bambino delle elementari è in grado di leggere le informazioni da un elenco telefonico.

Un modello più completo è invece il *Pagine Gialle*, che consente di catalogare i nominativi per categorie. In questo modo è possibile cercare un elettricista senza conoscerne il nominativo. In questo caso la struttura di organizzazione cambia e viene introdotto il concetto di categorie e macro-categorie, come ad esempio la categoria *Caldaie* può trovarsi all'interno della macro-categoria *Impianti*. Il modello semplificato di questo tipo di elenche è: *provincia*, *macro-categoria*, *categoria*, *nome attività*.

X.500 sono una serie di elenchi, standardizzati dall'ISO e organizzati per contenere informazioni su persone e risorse globali. La sua struttura, le categorie e gli attributi sono rigidamente prestabiliti e specificati.

4.1.1 Architettura di X.500

X.500 è una applicazione distribuita per la consultazione e la gestione di una base di dati strutturata ad albero denominata /indexDIB: Directory Information BaseDIB (Directory Information Base). Prevede due applicazioni:

DSA, che sta a significare Directory Service Agent, che opera sul server e DUA, Directory User Agents, che viene eseguito sui clients.

Ogni nodo del DIB è identificato univocamente da un *Distinguished Name*, che è il percorso compiuto dalla radice per giungere al nodo desiderato. Considerando l'esempio mostrato in figura 4.1 il Distinguished Name (dn) del nodo contenente i dati di Mario Rossi è: dc=it, dc=test, ou=people, ou=Vendite, cn=Mario Rossi.

La struttura ad albero è stata scelta poichè X.500 è utilizzato per frequenti letture, fatte in modo molto rapido, e limitate modifiche e inserimendo dei dati, per le quali è richiesto maggiore tempo di elaborazione.

Anche ciascuna *entry* ha gli attributi collezionati su una struttura ad albero, come mostrato in figura 4.2.

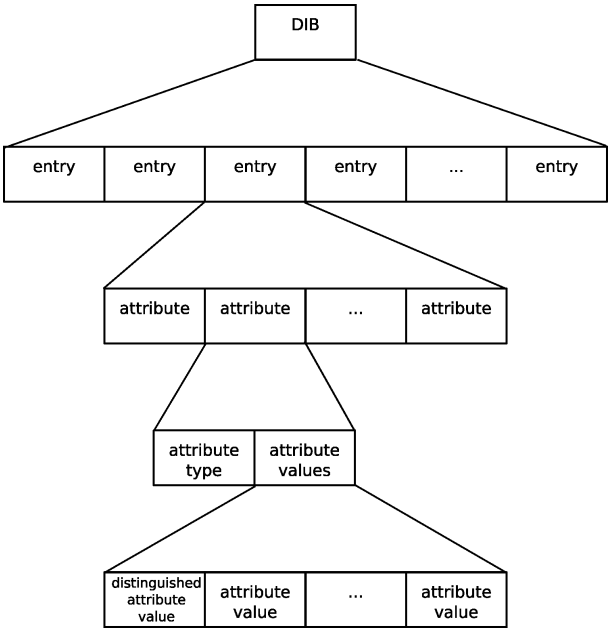


Figura 4.2: Struttura di una entry X.500

Se ciascuna entry ha il suo *distinguished name* (DN) che è il percorso completo che parte dalla radice dell'albero, il *relative distinguished name* (RDN), che è il valore che la identifica univocamente, è dato da uno o più suoi attributi figli. Ad esempio, se la nostra entry rappresenta una persona fisica,

potremmo identificare quella persona dal nominativo, da un identificativo o dal tipo di appartenenza.

4.1.2 Il servizio X.500

I DUA possono accedere al DSA tramite autenticazione ed effettuare le seguenti operazioni:

- Effettuare interrogazioni sulla directory
  Operazioni di read, compare, list, search.
- Effettuare modifiche sulla directory
  Operazioni di add, remove, modify delle entry e modify RDN

Sono anche disponibili dei *Service Control* che consentono di impostare:

- timeout,
- limiti di dimensione del risultato
- scopi di ricerca
- priorità delle richieste
- impostazione degli alias
- Operazioni distribuite

4.2 LDAP

LDAP è stato sviluppato per offrire uno strumento più leggero e flessibile per l'accesso alle directory X.500, ma anche a directory non più standardizzate e personalizzabili in base alle esigenze delle diverse realtà. A differenza dei database tradizionali è organizzato, come visto, in una struttura gerarchica e non in tabelle. Sempre rispetto X.500, ldap si differenzia sul fatto che si basa sul protocollo TCP/IP invece che sullo stack ISO/OSI; inoltre i dati sono costituiti da stringhe e non binari.

4.2.1 Elementi di LDAP

Se nello standard X.500 la radice dell'albero andava espressa nella forma:

```
o=testcompany, c=IT
```

dove il primo termine indica l'organizzazione (*O*) e il secondo la nazione (*C* di Country), ora è possibile specificare la radice anche nel seguente modo:

```
dc=testcompany, dc=it
```

oppure nel modo semplificato:

```
dc=testcompany.it
```

Questa nuova formattazione è molto simile a quella dei DNS, al punto che qualcuno sta pensando di rimpiazzare la tecnologia che sta dietro la risoluzione dei nomi di dominio con una struttura LDAP.

Tipicamente al di sotto del nodo che specifica la radice si posizionano dei nodi di *Organizational Unit* (ou) che consentono di dividere la struttura dati in sottogruppi. Nella figura 4.1 a pagina 47 sono state utilizzate divere *ou* per dividere, ad esempio, i dipendenti del settore vendite dai dirigenti.

Ci sono quindi i nodi contententi le informazioni finali, che possono trattarsi di dati di persone fisiche, dispositivi, strutture. A cui ci si riferirà utilizzando l'attributo *Common Name* (cn), ma anche in base ad altri parametri, quali ad esempio lo *User ID*, in base alle definizioni fornite dagli *schemi*.

4.2.2 Schemi di LDAP

Tutti i dati che compongono l'albero LDAP sono descritti negli schemi, in cui, per ogni oggetto, vengono definiti i campi che lo compongono ed i tipi di dati.
Gli *attributi* sono definiti in uno schema nel seguente modo:

```
attributetype ( 2.5.4.3 NAME ( 'cn' 'commonName' )
       DESC 'RFC2256: common name(s) for which the entity is known by'
       SUP name )
attributetype ( 2.5.4.4 NAME ( 'sn' 'surname' )
        DESC 'RFC2256: last name(s) for which the entity is known by'
        SUP name )
attributetype ( 2.5.4.35 NAME 'userPassword'
       DESC 'RFC2256/2307: password of user'
       EQUALITY octetStringMatch
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.40{128} )
attributetype ( 2.5.4.20 NAME 'telephoneNumber'
        DESC 'RFC2256: Telephone Number'
        EQUALITY telephoneNumberMatch
        SUBSTR telephoneNumberSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.50{32} )
```

Gli attributi sono raggruppati dagli *oggetti*, la cui definizione è formulata come segue:

```
objectclass ( 2.5.6.6 NAME 'person'
        DESC 'RFC2256: a person'
        SUP top STRUCTURAL
        MUST ( sn $ cn )
        MAY ( userPassword $ telephoneNumber ) )
```

Gli schemi principali sono definiti formalmente in diverse RFC, ma è tuttavia possibile definire degli schemi personalizzati in base alle particolari esigenze degli sviluppatori.

4.2.3 Architettura di LDAP

LDAP, come X.500, prevede un modello client-server per l'esecuzione delle operazioni, l'accesso ai dati contenuti nella directory può avvenire tramite autenticazione. LDAP supporta la replicazione e la possibilità di reindirizzare il client verso un altro server.

Nella figura 4.3 è schematizzato un sistema ad alta efficienza che prevede la replica del server principale su un cluster load balancing, su cui sono effettuate tutte le richieste dei client.

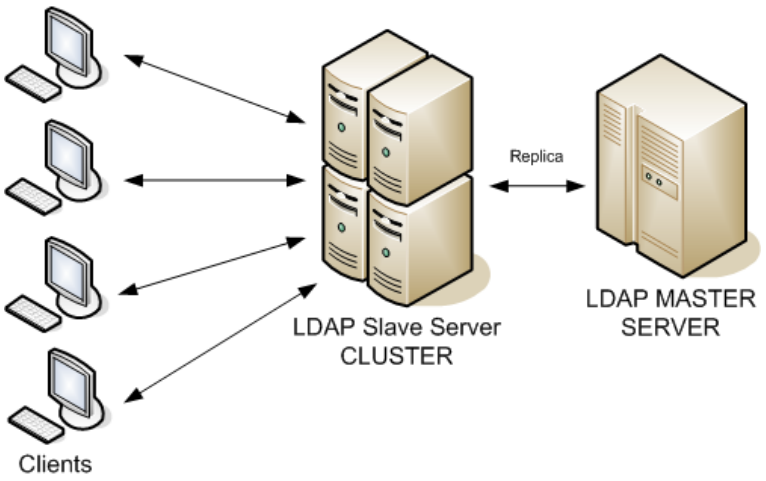


Figura 4.3: Esempio di replica LDAP

4.3 OpenLDAP

OpenLDAP è l'implementazione Open Source di LDAP disponibile in tutte le distribuzioni Linux. È sviluppato da una apposita fondazione ed è liberamente scaricabile all'indirizzo http://www.openldap.org. OpenLDAP supporta la distribuzione di carichi di lavoro, fornisce risposte in breve tempo e permette la replica delle informazioni.

4.3.1 Installazione e configurazione

OpenLDAP è disponibile precompilato per le principali distribuzioni linux, nonchè in codice sorgente. L'installazione in ambiente debian è, ancora una volta, molto semplice:

```
# apt-get install openldap
```

L'installazione dai sorgenti segue l'approccio standard:

```
# tar xfvz openldap.x.y.z.tar.gz
# cd openldap.x.y.z
# ./configure
```

```
# make depend
# make
# make install
```

Per aggiungere funzionalità a OpenLDAP, quali cifratura delle connessioni, è necessario installare anche le librerie OpenSSL-TSL (www.openssl.org).

Una volta terminata l'installazione avremo a disposizione i seguenti comandi:

slapd è il server che gestice le richieste dei client. Può essere in esecuzione su più macchine della stessa rete per aumentare l'efficienza.

slurpd è l'agente di replicazione tra i diversi server slapd.

Il file più importante per la configurazione di OpenLDAP è *slapd.conf*, che contiene tutte le informazioni di configurazione del server LDAP, ma anche le impostazioni per la replica.

Altri programmi di utilità sono disponibili per OpenLDAP, tra cui:

slapadd è usato per aggiungere entry specificate in formato *LDIF* nel database gestito da slapd.

slapcat consente di estrapolare i dati contenuti nella directory LDAP e salvarli in un file in formato LDIF. Molto utile per effettuare backup e migrazioni dei dati.

slapdn consente di verificare la conformità degli schemi inclusi nel file slapd.conf.

slapindex è usato per rigenerare gli indici della directory LDAP utilizzando i dati correntemente contenuti.

slappasswd consente di generare un hash di una stringa inserita per essere utilizzata come password, inserendolo in un file LDIF. Supporta diversi schemi di hashing: CRYPT, MD5, SMD5, SSHA (default) e SHA.

slaptest consente di verificare la correttezza del file di configurazione slapd.conf.

Ulteriori utility sono disponibili. Queste, però, sfruttano il protocollo LDAP per manipolare i dati contenuti nella directory LDAP. Vale la pena di presentare:

ldapadd tool per inserire una entry in un database LDAP.

ldapdelete tool per eliminare una entry.

ldapmodify consente di modificare il contenuto di una entry.

ldappasswd per cambiare la password di una determinata entry.

ldapsearch consente di interrogare la directory LDAP.

4.3.2 Il formato LDIF

LDIF è un acronimo di *LDAP Data Interchange Format*, usato da OpenLDAP per rappresentare le entry del database in formato testuale e ben leggibile.

Ciascun record è rappresentato come un gruppo di attributi. In un file, per separare un record da un altro è sufficiente lasciare una riga vuota. Il formato di una entry è:

```
Dn:<distinguished name>
<attrdescr>:<attrvalue>
<attrdescr>:<attrvalue>
...
```

Nel caso il valore di un attributo sia preceduto da "::" esso sarà rappresentato usando la codifica base64 ASCII.

Un esempio di file LDIF è il seguente:

```
dn: dc=testdomain,dc=it
dc: testdomain
o: testdomain.com
objectClass: top
objectClass: dcObject
```

```
objectClass: organization
structuralObjectClass: organization

dn: ou=people,dc=testdomain,dc=it
ou: people
objectClass: top
objectClass: organizationalUnit
structuralObjectClass: organizationalUnit
```

4.3.3 Il file slapd.conf

Slapd.conf è il file di configurazione del server slapd. È diviso in tre sezioni:

1. Direttive globali
2. Specifiche Backend
3. Specifiche Database

Direttive globali

In questa sezione sono inserite le direttive per configurare il funzionamento del servizio. Sono impostati i path dove trovare i moduli, impostazioni di log e, principalmente, l'inclusione degli schemi e le acl (access control list) per regolamentare l'accesso alla directory.

OpenLDAP viene fornito con diversi schemi, che contengono la definizione degli attributi necessari per le comuni operazioni di autenticazione.

L'inclusione degli schemi avviene con la seguente sintassi:

```
include         /etc/openldap/schema/core.schema
include         /etc/openldap/schema/cosine.schema
include         /etc/openldap/schema/nis.schema
include         /etc/openldap/schema/inetorgperson.schema
```

Un altro schema molto importate viene distribuito insieme al software samba (samba.schema) ed un altro con freeradius (RADIUS-LDAPv3.schema).

Questi due schemi offrono la possibilità di impiegare attributi utilissimi nel nostro caso, dove vogliamo implementare un sistema di controllo degli accessi basato su PPPoE, freeRADIUS e openLDAP.

Altro discorso va fatto per le acl. Con esse si stabilisce chi può accedere a determinati attributi e quali operazionio può compiere. Segue un esempio di acl:

```
access to attrs=userPassword
        by dn="cn=admin,dc=testdomain,dc=it" write
        by anonymous auth
        by self write
        by * none

access to dn.base="" by * read
```

Questa impostazione di acl è molto permissiva. Infatti chiunque può accedere a tutti i dati contenuti nella directory, ad eccezione dell'attributo *userPassword* cui può accedere in scrittura solo l'amministratore e il proprietario.

Una migliore impostazione potrebbe prevedere l'impiego di un utente chiamato, ad esempio, *ldapUser* che è autorizzato ad effetture le query sulla struttura LDAP, ma non a visualizzare le password. L'utente admin e ciascun singolo proprietario sono gli unici autorizzati a modificare le entry:

```
access to attrs=userPassword
        by dn="cn=admin,dc=testdomain,dc=it" write
        by self write
        by * none
access to attrs=sambaLMPassword
        by dn="cn=admin,dc=testdomain,dc=it" write
        by self write
        by * none
access to attrs=sambaNTPassword
        by dn="cn=admin,dc=testdomain,dc=it" write
        by self write
```

```
        by * none
access to dn.base="" by dn="cn=admin,dc=testdomain,dc=it" write
        by dn="cn=ldapUser,dc=testdomain,dc=it" read
by self write
by * none
```

Specifiche Backend

Il backend specifica con quale struttura dati saranno archiviati i dati. Ne esistono una grande quantità, come mostrato in tabella 4.1.

bdb	Berkeley DB Transactional backend
dnssrv	DNS SRV Backend
ldap	LDAP (proxy) Backend
ldbm	Lightweight DBM Backend
meta	Meta directory Backend
monitor	Monitor Backend
passwd	Accesso read-only a /etc/passwd
perl	Perl Programmable Backend
shell	Backend per l'uso di programmi esterni
sql	Backend per l'utilizzo del linguaggio SQL

Tabella 4.1: Tipi di Backend

Il backend predefinito è *ldbm*, mentre quello più utilizzato è, probabilmente, *bdb*. Utilizzando il backend *proxy* il server funziona in modalità proxy.

Il backend è specificato con una riga di configurazione simile alla seguente:

```
backend         bdb
```

Specifiche Database

Questa sezione contiene tutte le impostazioni di configurazione del database, e varia in base al tipo di backend scelto. Le opzioni più importanti di configurazione sono:

database *type* Indica l'inizio di una dichiarazione di istanza di un database, dove *type* è uno dei backend disponibili.

directory *path* Specifica il percorso in cui si trovano fisicamente i files utilizzati per archiviare il contenuto della directory.

suffix *dn* Specifica il base dn della directory. Ad esempio può essere: dc=testdomain, dc=it.

passowrdhash *hash* Specifica l'algoritmo con cui sarà cifrato l'attributo *userPassword*.

4.4 Replica di openLDAP

Il programma *slurpd* è l'agente di replicazione della directory LDAP. Viene usato per replicare le modifiche apportate sul *Master* ai vari *Slave*. Questo modello, che può essere implementato come mostrato in figura 4.3 a pagina 53, è impiegato quando il numero di client, o di interrogazioni, è particolarmente elevato.

Il file *slapd.conf* del Master deve contenere le seguenti direttive:

```
replica host=slave.testdomain.it:389
   binddn="cn=replicator,dc=testdomain,dc=it
   bindmethod=simple
   credential=password_in_chiaro
```

In questo modo la directory del Master sarà replicata sullo Slave, che dovrà avere un utente *replicator* che possa accedere in scrittura su tutte le entry.

Un parametro utile che può essere impiegato nel file di configurazione del Master è *replogfile* che consente di specificare un file dove saranno memorizzate, in formato LDIF, tutte le operazioni effettuate in replica.

La configurazione dello slave è differente. Alcuni dei parametri utilizzati sono:

```
backend=ldap
updatedn="cn=replicator,dc=testdomain,dc=it"
updateref ldap://master.testdomain.it/
```

4.5 Connessioni sicure con TSL/SSL

Il protocollo LDAP prevede connessioni TCP in chiaro sulla porta 389. Questo costituisce non pochi problemi di sicurezza nel caso il programma client non sia in esecuzione sulla stessa macchina, o nel caso di repliche.

Per ovviare a questo problema è necessario cifrare la comunicazione. I server e i client openLDAP possono, infatti, usare il Transport Layer Security (TLS), una evoluzione di SSL, per cifrare le comunicazioni tra gli host a livello TCP.

Il TSL prevede l'utilizzo di certificati, che possono essere presi da terze parti o generati in proprio con openssl.

Per impiegare il TSL è necessario inserire, nel file *slapd.conf*

```
TLSCipherSuite HIGH:MEDIUM:+SSLv2
TLSCertificateFile /etc/openldap/certs/slapd.pem
TLSCertificateKeyFile /etc/openldap/certs/slapd.key
```

4.6 Popolazione e gestione di una directory

Esistono diversi modi e strumenti per la popolazione di una directory ldap. Inizialmente era necessario creare dei file LDIF e farli elaborare dal comando *ldapadd*, o dal comando *slapcat*.

Con la maggiore diffusione di LDAP sono comparsi strumenti sempre più *user friendly*, flessibili e potenti. Alcuni di loro sono web-based, utilizzabili comodamente anche da remoto. Altri sono dei client LDAP con un interfaccia più o meno graziosa.

LDAP sta attraendo l'attenzione di un numero sempre maggiore di sviluppatori e si attendono strumenti sempre più interessanti. Procederemo con una breve panoramica di quelli che sono alcuni strumenti oggi disponibili.

4.6.1 smbldap-tools

Si tratta di una collezione di tools scritti in Perl, eseguiti in linea di comando. Sono progettati per essere utilizzati su una directory LDAP configurata per

usare lo schema *samba.schema* e quindi utilizzabili per gestire utenti di una rete Microsoft.

Necessita di una configurazione iniziale ed hanno il grande vantaggio di salvare la stessa password negli attribuiti em userPassword, *sambaNTPassword* e *sambaLMPassword*, che sono generate in modo diverso.

I principali comandi messi a disposizione dal pacchetto sono:

smbldap-useradd consente di aggiungere un utente (o un host).

smbldap-passwd consente di cambiare la password di un utente.

smbldap-userdel per cancellare un utente (o un host).

smbldap-usermod modifica le informazioni di un utente.

smbldap-usershow mostra le informazioni di un utente.

I *smbldap-tools* sono strumenti semplici e veloci da utilizzare. Ideali per realizzare degli script strutturati o per inserimenti automatizzati.

4.6.2 LDAP Browser

Si tratta di un applet java, quindi eseguibile su tutte le piattaforme, che consente la completa gestione di una directory LDAP. È un tool che inizia a mostrare i segni dell'età, ma è ancora da preferire per chi desidera uno strumento estremamente flessibile e rapido.

In figura 4.4 è mostrata una schermata del programma *LDAP Browser*

4.6.3 LUMA

Luma è un interfaccia grafica per l'accesso e l'amministrazione di directory LDAP. Scritto in Python, usa le librerie PyQT e python-ldap. Il programma è funzionale, ma è probabilmente troppo macchinoso per quanto riguarda l'amministrazione degli utenti. Uno screenshot è mostrato in figura 4.5

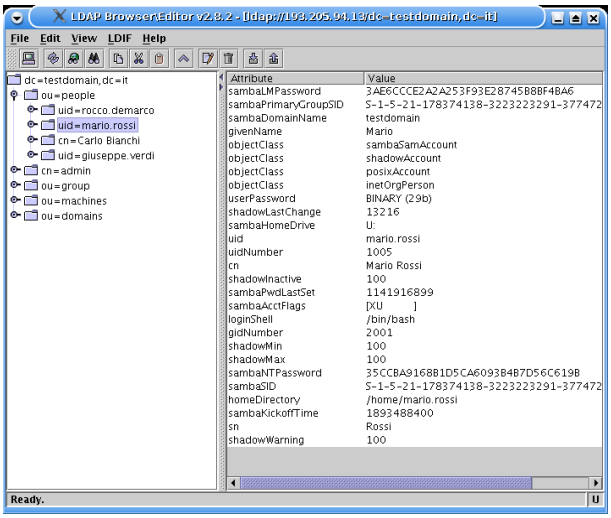


Figura 4.4: LDAP Browser

4.6.4 Interfaccie web-based

Esistono diverse interfacce web-based per la gestione delle directory LDAP. Sono la soluzione che preferisco per la completezza della gestione e per la flessibilità.

Tra quelli che ho provato i prodotti migliori sono *PhpLDAPAdmin* e ldap account manager. Tra i due il secondo è quello che mi piace di più, ma sono sostanzialmente uguali.

Ritengo la gestione degli utenti attraverso interfacce web-based una buona soluzione, anche dal punto di vista della sicurezza. Infatti impiegando *Apache ssl* come server http, tutte le informazioni scambiate tra il browser ed il server saranno cifrate. Se poi sul server vengono impostate delle valide politiche di firewalling, ovvero restringere l'accesso solo da determinate macchine, la sicurezza è molto alta.

Utilizzo *Ldap Account Manager* per gestire directory con centinaia di entry, e sono soddisfatto, anche per la possibilità di generare dei pdf. L'unica pecca è dovuta al fatto che non è ancora disponibile un modulo che aggiunga gli attributi RADIUS, cosa già presente su *PhpLDAPAdmin*. Nelle figure 4.6 e 4.7 sono mostrati questi due applicativi.

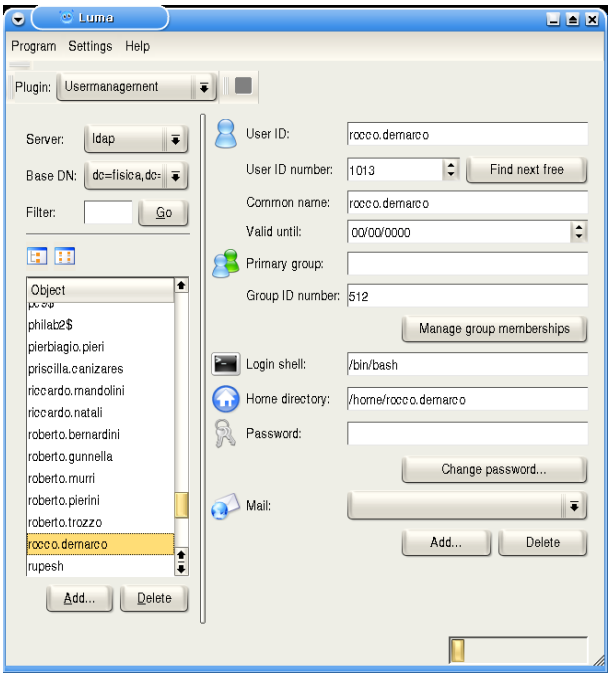


Figura 4.5: Luma LDAP GUI

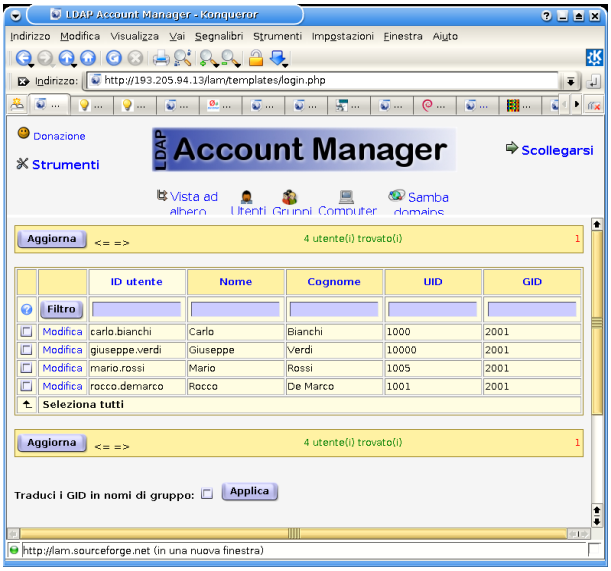


Figura 4.6: Ldap Account Manager

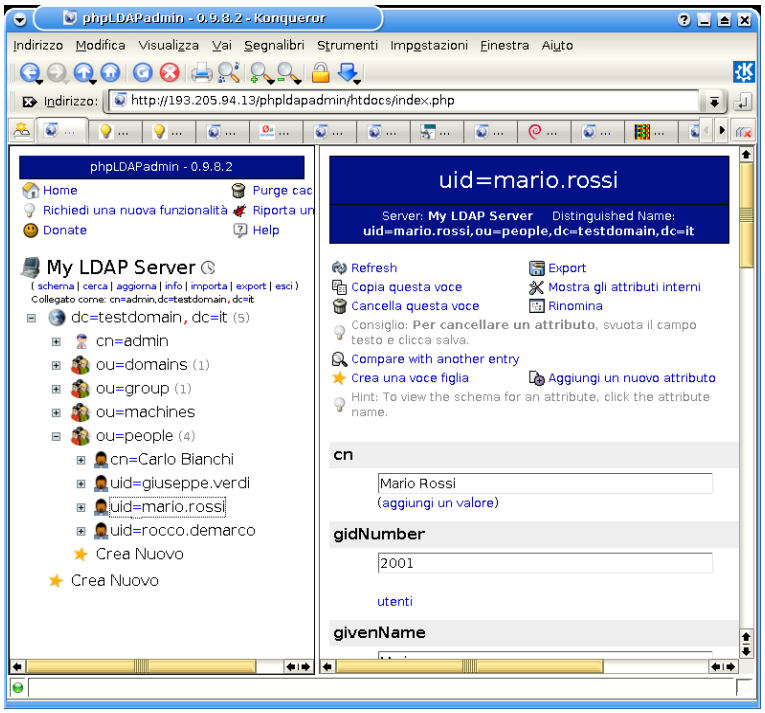


Figura 4.7: phpLDAPAdmin

Capitolo 5

Autenticare con openLDAP

Il presupposto prefissato in partenza era quello di dare a “Cesare” una sola password. Una sola password per tutti i servizi. Con LDAP questo è oggi possibile, anche se non è l’unica soluzione a disposizione, ma LDAP è sicuramente lo strumento più espandibile sul panorama. Espandibile al punto di inglobare qualsiasi altro sistema di autenticazione, kerberos incluso.

Questo capitolo vuole illustrare come utilizzare LDAP per amministrare l’accesso ai principali servizi disponibili, in modo che l’utente abbia sempre lo stesso username e sempre la stessa password da utilizzare in tutte le circostanze.

Non è una esagerazione dire che LDAP sarà sempre più importante e diffuso nel futuro.

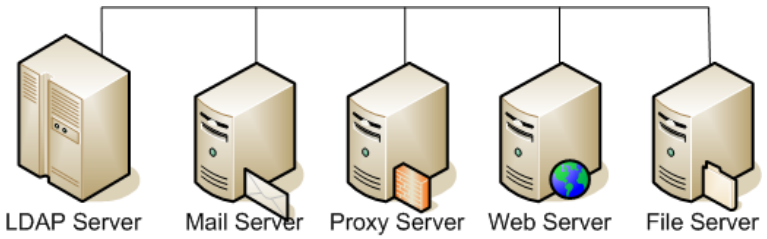


Figura 5.1: LDAP per autenticare diversi servizi

5.1 Controllo dei login

In numerose realtà esiste il problema di restringere l'accesso agli elaboratori unicamente a persone abilitate: aziende, laboratori didattici, internet point, e così via. Restrigere l'accesso, ma anche decidere se centralizzare i dati su una determinata macchina, se offrire la possibilità di stampare ed eventualmente quali restrizioni adottare.

Il panorama odierno vede l'impiego di due sistemi operativi in particolare, Microsoft Windows ed il Linux, per i quali valuteremo i distinti casi.

5.1.1 Accesso a Windows

Per la piattaforma Microsoft Windows da alcuni lustri è disponibile la famigerata "rete Microsoft" composta da domini NT su cui un server, il *Primary Domain Controller*, ha a disposizione l'elenco degli utenti e delle workstation e le politiche con cui i primi accedono alle seconde.

L'evoluzione delle reti Microsoft è stata *Active Directory* che altro non è che una implementazione di LDAP. Il funzionamento di AD è simile a quello delle prime, aggiungendo nuove features, ma offrendo sostanzialmente lo stesso scopo: gestione di utenti e macchine.

L'implementazione OpenSource del protocollo SMB (usato da AD e le reti Microsoft) è il progetto SAMBA (www.samba.org). Da diversi anni è possibile implementare domini NT dove il PDC è una macchina Linux su cui è in esecuzione samba.

Samba offre molte delle opzioni fornita dalla piattaforma Microsoft: roaming dei profili, gestione delle quote, degli accessi sulle workstation, condivisioni, stampanti. I client windows non notano particolari differenze, o mostrano problemi, avendo come proprio PDC un server linux.

Molta documentazione è disponibile per spiegare come configurare un server samba PDC, mentre in questo contesto di vuole illustrare come utilizzare LDAP per consentire agli utenti windows di autenticarsi tramite samba.

Sostanzialmente è necessario configurare un server samba PDC per gestire un dominio NT. Samba tipicamente usa il file smbpasswd per la gestione degli

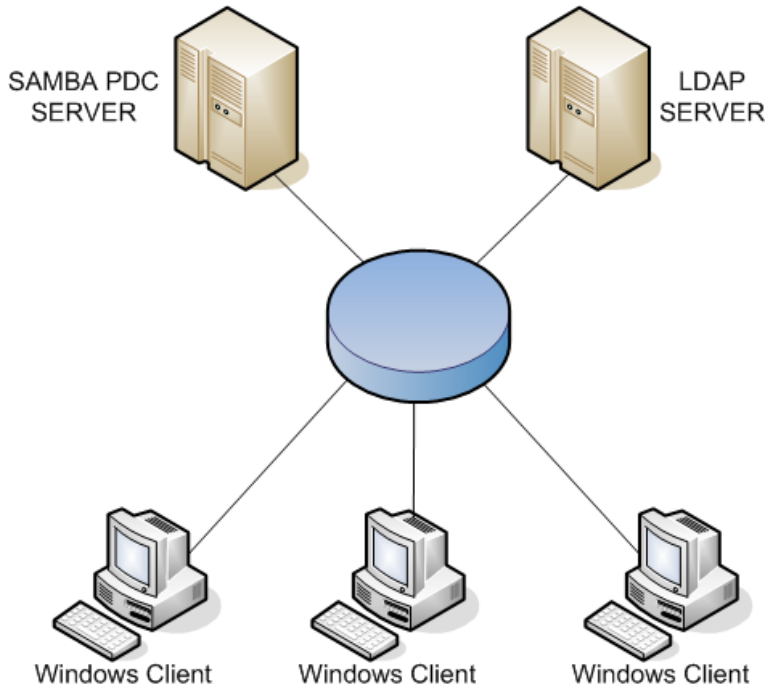


Figura 5.2: Samba PDC e LDAP

utenti, noi invece ci collegheremo ad un server LDAP, su cui è necessario che sia incluso nella configurazione il file *samba.schema*, che fornisce -tra gli altri- gli attibuti *sambaNTPassword* e *sambaLMPassword*. Per fare questo è necessario che samba sia compilato con il supporto ldap, consultare la documentazione della propria distribuzione per verificarlo.

Il file *smb.conf* del server PDC dovrebbe contenere una configurazione simile alla seguente:

```
[global]
   ...
   workgroup = TESTDOMAIN
   security = user
   domain master = yes
   ...
   passdb backend = ldapsam:ldap://127.0.0.1:389
   ldap admin dn = cn=admin,dc=testdomain,dc=it
```

```
   ldap user suffix = ou=people
   ldap group suffix = ou=groups
   ldap machine suffix = ou=hosts
   ldap passwd sync = Yes
   ldap suffix = dc=testdomain,dc=it
   ldap ssl = no
   ...
[homes]
   comment = Home Directories
   read only = No
   create mask = 0700
   directory mask = 0700
   browseable = No

[profiles]
   path = /etc/samba/profiles
   read only = No
   create mask = 0600
   directory mask = 0700
...
```

In questo caso il PDC del dominio TESTDOMAIN andrà a leggere le informazioni di accesso su un server openLDAP che è in esecuzione sulla stessa macchina.

La password di admin va impostata con il comando

```
# smbpasswd -w
```

La configurazione dei client è decisamente semplice: dopo aver annesso il computer al dominio (in questo caso TESTDOMAIN) ciascun utente potrà accedere sulla postazione usando lo username e la password (*sambaNTPassword*) conservati nella directory LDAP.
Una entry di un utente generico che usa LDAP per accedere a macchine Windows su dominio samba deve avere la seguente collezione di attributi:

```
dn: uid=mario.rossi,ou=people,dc=testdomain,dc=it
```

```
uid: mario.rossi
sambaSID: S-1-5-21-178374138-3223223291-377472742-3010
cn: Mario Rossi
sambaKickoffTime: 1893488400
sambaAcctFlags: [XU         ]
sambaHomeDrive: U:
sambaPrimaryGroupSID: S-1-5-21-178374138-3223223291-377472742-4003
sambaDomainName: testdomain
sambaProfilePath: \\PDC\profiles\mario.rossi
sambaHomePath: \\PDC\home\mario.rossi
objectClass: sambaSamAccount
objectClass: shadowAccount
objectClass: posixAccount
objectClass: inetOrgPerson
uidNumber: 1005
gidNumber: 2001
homeDirectory: /home/mario.rossi
loginShell: /bin/bash
givenName: Mario
sn: Rossi
structuralObjectClass: inetOrgPerson
shadowMin: 100
shadowMax: 100
shadowWarning: 100
shadowInactive: 100
sambaLMPassword: 3AE6CCCE2A2A253F93E28745B8BF4BA6
sambaNTPassword: 35CCBA9168B1D5CA6093B4B7D56C619B
sambaPwdLastSet: 1141916899
shadowLastChange: 13216
userPassword:: e01ENX1GNXJVWEd6aXk1ZlBFQ25pRWdSdWdRPT0=
```

Un valido strumento per gestire utenti con il sistema samba + openLDAP sono le smbldap-utils, di cui si è parlato nel capitolo precedente.

5.1.2 Accesso su Linux

Da diversi anni linux gestisce l'autenticazione con degli appositi moduli, chiamati PAM (Pluggable Authentication Module). Tra essi è presente il modulo *pam_ldap* che consente di autenticare gli utenti con i dati presenti nella directory LDAP. Un altro software che deve essere installato è il modulo NSS (Name Service Switch): libnss-ldap, che consente di associare e considerare locali gli utenti presenti sulla directory.

Concedere l'accesso ad una macchina linux con utenti posti su un server ldap

pone il problema di dove salvare i dati dell'utente. Esistono principalmente due soluzioni, ciascuna con dei pro o dei contro:

- Montare un filesystem remoto (con samba o nfs): può esporre a problemi di sicurezza e performances scadenti.
- Creare on-demand una homedirectory sulla macchina locale: esiste un modulo chiamato *mkhomedir*, ma causa distribuzione dei dati su più macchine.

Personalmente preferisco usare contemporaneamente entrambi, in un modo riarrangiato. L'utente normalmente lavora sul computer locale, ma ha una directory montata sul disco di rete su cui può deliberatamente trasferire documenti di cui vuole poter disporre in altre postazioni. Questo limita i trasferimenti di files (sapete che significa trasferire un profilo da 5GB?) ed evita problemi in caso di intasamento della rete.

Per configurare il client con lo scopo di sfruttare l'autenticazione LDAP bisogna modificare diversi files: *pam_ldap.conf*, *nsswitch.conf* e diversi files nella directory */etc/pam.d*.

Configurare pam_ldap.conf

Questo file permette di configurare il PAM-LDAP per consentire l'accesso in locale di utenti registrati su un server LDAP. Tra le varie opzioni di configurazione devono essere presenti, ad esempio:

```
host 127.0.0.1
base dc=testdomain,dc=it
```

Configurare i PAM

Nella directory */etc/pam.d* sono presenti diversi file di autenticazione, il cui nome rimanda al servizio che espletano. In una distribuzione Debian GNU/Linux 3.1 sono disponibili:

```
atd               groupadd      samba
chage             groupdel      ssh
chfn              groupmod      ssh.dpkg-dist
chsh              kcheckpass    su
common-account    kdm           sudo
common-auth       kdm-np        suexec-apache
common-password   kscreensaver  useradd
common-session    login         userdel
cron              newusers      usermod
cupsys            other         webmin
cvs               passwd        wu-ftpd
gdm               ppp           xscreensaver
gdm-autologin
```

Probabilmente in qualsiasi distribuzione l'elenco appare differente. Per questa configurazione è consigliabile consultare la documentazione relativa a ciascuna distribuzione.

Per poter autenticare gli utenti con ldap vanno modificati tutti i file necessari (ad esempio *login* ed eventualmente *ssh* ed inserite le righe:

```
auth     sufficient pam_ldap.so
account  sufficient pam_ldap.so
password sufficient pam_ldap.so
session  sufficient pam_ldap.so
```

Configurare nsswitch.conf

Il Name Service Switch si occupa di convertire UID in username, cosa necessaria per un normale funzionamento di un sistema linux. Ad esempio, viene utilizzato quando andiamo ad eseguire il comando *ls -l*, dove sono elencati i proprietari di ciascun file.

Il file di configurazione, *nsswitch.conf*, deve essere configurato in questo modo:

```
passwd:    compat ldap
```

```
group:      compat ldap
shadow:     compat ldap
```

Per aumentare le prestazioni, considerando che le richieste sarebbero diverse decine al secondo, si consiglia di installare nscd (*Name Server Cache Daemon*).

5.2 LDAP per autenticare il web

Molti gestionali del passato sono oggi diventate applicazioni web-based. Realizzare un servizio via web offre numerosi vantaggi:

- Supera l'eterogeneità del "parco macchine", dato che in molte realtà gli utenti utilizzano sistemi operativi diversi che non consentono, escludendo java, di realizzare applicazioni che possano essere eseguite ovunque. Tutti i sistemi operativi, nonchè numerosi dispositivi di telefonia mobile, offrono un web browser.
- Consente di semplificare la gestione, avendo un solo elaboratore su cui effettuare controlli e backup.
- Offre la possibilità di erogare i servizi ovunque: gli utenti potranno accedere all'applicativo anche da casa, o dall'estero durante le missioni.
- Le applicazioni web-based sono facilmente scalabili, sia in termini di servizi che di prestazioni (basta acquistare harware più potente o implementare un cluster *load balancing*).
- L'interfaccia grafica web-based è intuitiva e semplice da utilizzare.

I difetti sono presenti e non sono assolutamente trascurabili:

- Il server che fornisce il servizio web-based è esposto a maggiori rischi di aggressione, soprattutto se eroga servizi all'esterno della rete locale.
- In caso di *fault* del server, o della rete, il servizio non è più disponibile per nessun utente. Cosa ovviabile predisponendo un backup server da rendere prontamente disponibile.

Tutti i servizi web-based dovrebbero essere erogati utilizzando connessioni SSL, in modo da ridurre il rischio che le informazioni trasmesse siano intercettate. Inoltre è necessario autenticare l'utente prima di accordare l'accesso al servizio.

Questa autenticazione è possibile grazie all'impiego del server http *apache-ssl*, il framework *php4*, la libreria *php4-ldap* e, ovviamente, *openLDAP*. Tralasceremo in questo contesto la configurazione dei singoli pacchetti, di cui si trova ampia documentazione su internet.

Per abilitare il supporto LDAP su php4 è necessario accertarsi che sia presente, nel file *php.ini*, la seguente riga:

```
extension=ldap.so
```

Se tutto è ben configurato sarà possibile interrogare la directory LDAP con delle pagine scritte in php. Vediamo un esempio di autenticazione:

```
$ldap_server="127.0.0.1";
$user="mario.rossi";
$pass="testpass";

$username="uid=$user,ou=people,dc=testdomain,dc=it";

// mi collego con il server ldap
$ds=@ldap_connect($ldap_server);
if ($ds)
{
   // se riesco ad accedere ho verificato username e password
   if(@ldap_bind($ds,$username,$pass))
   {
     // ok, l'utente è verificato
     // seguono le routine di impostazione dei cookie
   } else {
       echo "<h2>Accesso NON riuscito</h2>";
   }
}
@ldap_unbind($ds);
```

Una volta effettuato il *bind* è possibile effettuare delle interrogazioni sulla directory LDAP, come l'esempio che segue:

```
$sr=ldap_search ($ds, "ou=people,dc=testdomain,dc=it", "uid=$user");
$entry = ldap_first_entry($ds, $sr); #get the result
$home = ldap_get_values($ds, $entry, "homeDirectory");
$desc = ldap_get_values($ds, $entry, "description");
$cn = ldap_get_values($ds, $entry, "cn");
$gecos = ldap_get_values($ds, $entry, "gecos");
$gid = ldap_get_values($ds, $entry, "gidNumber");
```

L'autenticazione tramite LDAP è disponibile anche per diversi CMS[1], tra cui *drupal*, ma anche molte applicazioni commerciali si muovono nella direzione dell'autenticazione tramite LDAP.

5.3 LDAP e freeRADIUS

OpenLDAP può essere utilizzato con soddisfazione per le fasi di *Autenticazione* e *Autorizzazione* di un server freeRADIUS.

La fase di autenticazione avviene tramite il controllo delle credenziali contenute negli attributi *uid* e *sambaNTPassword*, come già discusso nel primo capitolo.

La gestione dell'autorizzazione è invece possibile grazie ad uno schema fornito con freeRADIUS, *RADIUS-LDAPv3.schema*, che consente di aggiungere un elenco di attributi utilizzabili per queste finalità. La pecca principale è la quasi totale mancanza di documentazione fornita con questo schema.

Il file RADIUS-LDAPv3.schema deve essere copiato nella directory contenenti gli schemi di LDAP, e incluso inserendo la seguente riga nella sezione globale del file *slapd.conf*:

```
include         /etc/openldap/schema/RADIUS-LDAPv3.schema
```

Gli attributi contenuti in questo schema sono 61, di cui tra i principali troviamo:

[1] Content Management System

radiusFramedIPAddress Indirizzo IP assegnato al client durante la connessione.

radiusFramedIPNetmask Netmask assegnata al client.

radiusFramedMTU Consente di impostare la MTU, ovvero la massima dimensione trasferibile in un pacchetto. Usando PPPoE dovrebbe essere impostata a 1490.

radiusFramedRoute Consente di impostare la tabella di routing del client.

radiusIdleTimeout Imposta il tempo di timeout: trascorso il tempo prefissato di inattività la connessione viene chiusa.

radiusLoginService Specifica il tipo di servizio accordato al client.

radiusSessionTimeout Specifica la durata massima, in secondi, di una sessione utente.

radiusSimultaneousUse Numero massimo di accessi simultanei consentiti.

Nel corso di queste pagine viene illustrato come autenticare gli utenti utilizzando freeRADIUS e openLDAP, tramite il protocollo MSCHAPv2. Questo protocollo, seppur molto diffuso, ha mostrato poteziali vulnerabilità che ne sconsigliano l'impiego in determinati casi.

Una valida alternativa è quella di impiegare il protocollo EAP/PEAP, che consiste sempre in una autenticazione tramite username e passoword con MSCHAPv2, ma sfrutta una crittografia più solida. FreeRADIUS consente di autenticare tramite EAP/PEAP, andando a leggere le informazioni utenti da una directory LDAP. Questo sistema può essere utilizzato anche per autenticare utenti tramite il protocollo 802.1X.

La configurazione di freeRADIUS richiede il possesso di certificati, che possono essere ottenuti da una PKI, o prodotti con openssl. Il file *eap.conf* nella cartella dei file di configurazione di freeRADIUS deve essere impostato in un modo simile al seguente:

```
default_eap_type = peap
```

```
...
  tls {
        # The private key password
        private_key_password = SecretKeyPass77
        # The private key
        private_key_file = ${raddbdir}/certs/cert-srv.pem
        #  Trusted Root CA list
        CA_file = ${raddbdir}/certs/demoCA/cacert.pem
        dh_file = ${raddbdir}/certs/dh
        random_file = /dev/urandom
        }
...
      peap {
        #  The tunneled EAP session needs a default
        #  EAP type, which is separate from the one for
        #  the non-tunneled EAP module.  Inside of the
        #  PEAP tunnel, we recommend using MS-CHAPv2,
        #  as that is the default type supported by
        #  Windows clients.
        default_eap_type = mschapv2
      }
```

5.4 LDAP e la posta elettronica

Uno dei servizi chiave da offrire agli utenti è quello di posta elettronica. Questo servizio richiede l'utilizzo di due serventi: il transport agent (server SMTP), che si occupa di recapitare il messaggio alla mailbox del destinatario, e il servizio di consulatazione della mailbox (POP3 o IMAP).

Una soluzione completa, ma commerciale, per la gestione della posta elettronica è il software *Communigate Pro*[2], che è implementato proprio con una directory LDAP. Anzi, questo software può essere utilizzato perfino come server LDAP.

[2]Communigate Pro è un software della Stalker Inc.

In ambiente Open Source i principali server SMTP offrono il supporto di LDAP. È il caso di alcuni dei software più diffusi: *exim*, *postfix* e *qmail*. Vedremo brevemente come è possibile configurare il secondo: *Postfix*.

Avendo a disposizione una versione di Postfix compilata con il supporto LDAP, bisogna apportare alcune semplici modifiche ai file di configurazione. Il file *main.cf* deve essere modificato come segue:

```
alias_maps=hash:/etc/aliases, ldap:/etc/postfix/ldap-aliases.cf
```

Il file *ldap-aliases.cf* conterrà, per esempio:

```
server_host = 127.0.0.1
search_base = dc=testdomain, dc=it
bind_dn = cn=postfixAgent, dc=testdomain, dc=it
bind_pw = topsecret
```

L'utente *postfixAgent* deve essere presente sulla directory LDAP e deve avere diritti per poter effettuare delle ricerche in essa.

Con questa semplice configurazione è possibile ricevere email in cui la mailbox è corrispondente allo *uid* presente sulla directory LDAP. Per maggiori informazioni si consiglia di consultare il sito www.postfix.org.

5.5 LDAP e SQUID

SQUID è uno dei più diffusi proxy server. I proxy server, nati per ridurre l'utilizzo della banda facendo caching delle pagine web, sono nel corso degli anni diventati strumenti di controllo e monitoraggio del traffico web. Nei suoi log è possibile recuperare informazioni sulle attività di web surfing degli utenti, ma anche di limitare l'accesso a internet sfruttando l'autenticazione.

Un internet point potrebbe essere strutturato in modo da avere un proxy server con due interfacce di rete: la prima verso la rete locale su cui si connettono i client, la seconda verso l'esterno. Per poter accedere all'esterno i client devono impostare il proprio browser per utilizzare il proxy server, che richiede l'autenticazione. Autenticazione che sarà accordata valutando

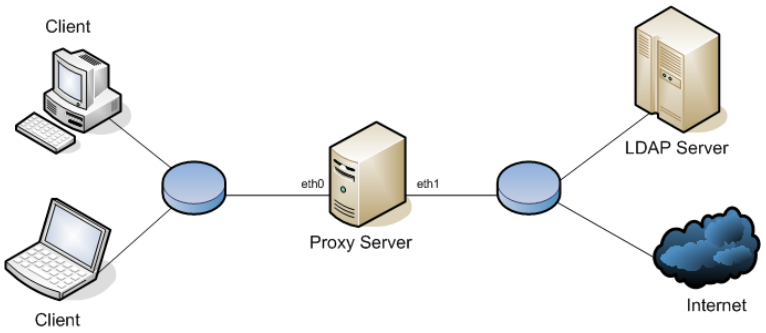


Figura 5.3: Controllo del traffico con squid e openLDAP

i dati degli utenti presenti su una directory ldap. Un esempio di questa architettura è mostrata in figura 5.3.

Squid utilizza un *helper* chiamato *squid_ldap_auth*, normalmente fornito con squid. La configurazione per utilizzarlo è decisamente facile e richiede l'inserimento di poche righe nel file di configurazione *squid.conf*

```
acl all src 192.168.1.0/24

auth_param basic program squid_ldap_auth \
   -b dc=testdomain,dc=it -f uid=%s 127.0.0.1
auth_param basic children 5
auth_param basic realm Squid proxy-caching web server
auth_param basic credentialsttl 30 minutes

acl password proxy_auth REQUIRED
http_access allow password all
```

In questo modo sarà consentito l'accesso alla rete esterna, per mezzo del proxy, solo agli utenti che forniscono credenziali valide.

Capitolo 6

PPPoE e Reti Wireless

Le reti wireless in pochi anni sono divenute un fenomeno con una elevatissima diffusione. I prodotti in commercio sono molto differenti e variano in campo d'azione, velocità, affidabilità e protocolli integrati.

Il dispositivo che consente di accedere via wireless ad una rete tradizionale si chiama *Access Point*, che normalmente si comporta come un HUB ethernet, dove gli utenti condividono la banda.

I motivi principali del successo delle reti wireless sono la semplicità d'installazione e la comodità di utilizzo. Laddove è necessario stendere diverse centinaia di cavo UTP, forare le mura, effettuare le interconnessioni, testare ciascun singolo cavo, con un paio di access point è possibile offrire connettività agli utenti. Utenti che con una semplice operazione di *Discovery* troveranno l'access point più vicino cui collegarsi.

Il prezzo da pagare è tuttavia elevato: la sicurezza costituisce un problema grave e le prestazioni diminuiscono al crescere del numero di utenti. Si, la sicurezza è un problema, perchè le onde radio, purtroppo, escono fuori dal nostro edificio e chiunque dall'esterno potrebbe accedere alla nostra rete wireless.

Esistono diversi metodi per difendersi, alcuni semplici da implementare, altri meno. Alcuni sono pure addizionabili, in modo da innalzare ulteriormente il livello di sicurezza.

In questo capitolo si daranno alcuni cenni sul funzionamento delle reti wireless e su alcuni metodi di protezione, ma la parte che sarà approfondita maggiormente sarà quella che vede l'utilizzo del PPPoE. Per ulteriori informazioni si rimanda ai testi citati nella bibliografia.

6.1 Architetture delle reti wireless

Scartando le architetture più elementari, è chiaro che un access point debba essere collocato dietro un firewall. Questo è il minimo per limitare i danni, o perlomeno restringerne la portata.

Un errore che viene normalmente commesso dai progettisti di rete meno esperti è quello di acquistare dispositivi con un elevato raggio di copertura. Il vantaggio è inesistente, visto che tali dispositivi costano il doppio o il triplo di access point con un breve raggio d'azione. In più fioccano difetti come la grandine: un elevato raggio di copertura espone maggiormente la WLAN al rischio di accessi non autorizzati, inoltre avere un elevato numero di utenti per dispositivo degenera le prestazioni.

Una soluzione valida è quella di prevedere la divisione dell'area da asservire in celle, dove installare, su ciascuna di esse, un access point. Una cella non dovrebbe contenere più di 10 utenti, e non dovrebbe avere una estensione superiore ai 20 metri, come mostrato in figura 6.1.

Vedremo in seguito che con l'impiego del PPPoE non sono necessari access point che offrano particolari servizi, ma vanno benissimo anche quelli per il mercato consumer. Sarebbe comunque meglio acquistare dispositivi che offrano il supporto dell'autenticazione 802.1X e WPA, anche se oggi si trovano sul mercato dei prodotti, come il DLINK DWL 2100 AP+ che costa intorno ai 60 euro.

6.1.1 Utilizzando un firewall

La nostra rete LAN dovrebbe essere separata da una rete wireless attraverso un firewall, con politiche più restrittive rispetto quelle adottate per gli utenti connessi via cavo.

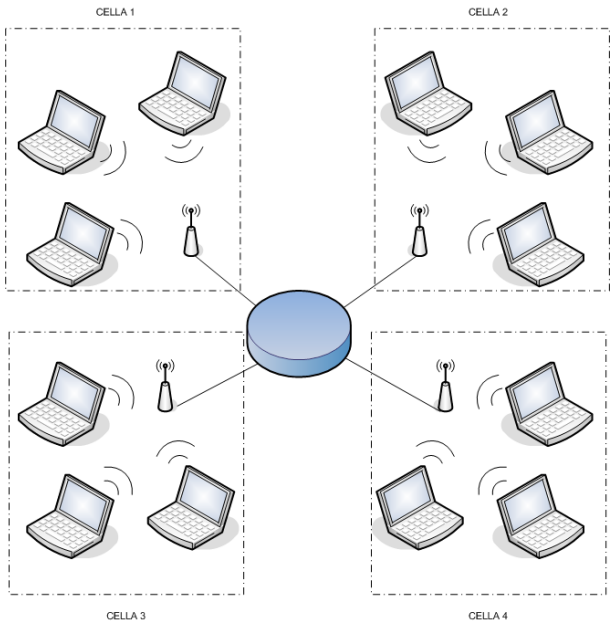


Figura 6.1: Esempio di rete wireless

Un metodo elementare per implementare un minimo di controllo degli accessi è quello di utilizzare un server *dhcp* chiuso in modo da associare un indirizzo IP ad un indirizzo MAC. Inoltre un filtro sugli indirizzi MAC potrebbe essere attivato su ciascun access point, per impedire l'accesso a chi non appare nella lista.

Ma è una protezione che non serve a niente. Basta utilizzare un programma come *kismet*, mostrato in figura 6.2, per intercettare le comunicazioni, carpire un MAC e assegnarlo sulla propria interfaccia wireless. A questo punto il nostro ospite si trova proiettato nella nostra rete, con la libertà di accedere alle nostre risorse e, tipicamente, sfruttare la connessione a internet.

Inoltre il traffico che avviene tra il client e l'access point è in chiaro e può essere facilmente intercettato. Il primo controrimedio potrebbe essere l'impiego delle chiavi WEP.

Il WEP (Wire Equivalent Protocol) è un protocollo inutile e può essere forzato in 15 minuti. Senza soffermarci più del dovuto, va detto che il WEP è un protocollo di crifratura delle connessioni wireless. Può essere a 64, 128 e 256 bit, ma la lunghezza della chiave non rende più sicuro questo protocollo.

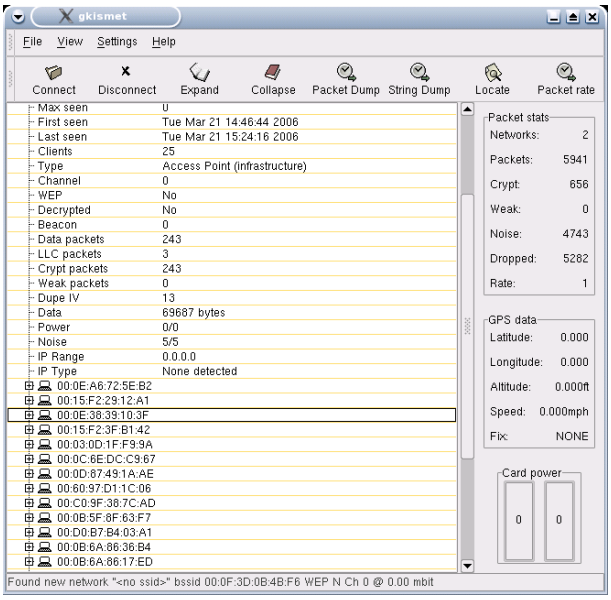


Figura 6.2: Kismet

Esistono alcuni tool, come *airsnort* (figura 6.3), che consentono di decifrare agevolmente la chiave wep.

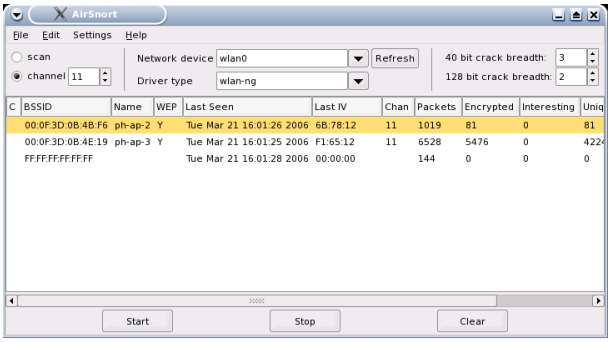


Figura 6.3: Airsnort

6.1.2 Utilizzando WPA e 802.1X

WPA, Wi-Fi Protected Access, è un protocollo di sicurezza per le reti wireless che implementa parte del protocollo IEEE 802.11i. I dati sono cifrati

utilizzando l'algoritmo RC4 con chiave a 128 bit e intestazione a 48 bit. I principali vantaggi sono:

- Le chiavi crittografiche sono scambiate periodicamente: Temporal Key Integrity Protocol (TKIP). Questo rende vano il tentativo di crittanalisi.
- Il WPA introduce notevoli miglioramenti nella gestione dell'integrità. Il CRC utilizzato dal WEP non era sicuro, era possibile modificare il messaggio mantenendo coerente in CRC anche senza conoscere la chiave.
- Possibilità di utilizzare il protocollo 802.1x per autenticare gli utenti.

Ha purtroppo degli svantaggi:

- Non tutto l'harware in circolazione supporta il protocollo WPA. Questo è un problema serio e difficilmente superabile.
- Richiede un supplicat che è spesso a pagamento.
- Non è possibile assegnare un indirizzo IP statico ad un utente.
- Sono noti dei problemi di fault degli access point in caso di attacco DOS.

Probabilmente il WPA e 802.1x sarà il migliore sistema di gestione delle reti wireless in futuro quando aumenterà l'harware che lo supporta e saranno disponibili dei supplicant per tutti i sistemi operativi senza spese aggiuntive.

Torneremo in seguito a discutere di questo argomento.

6.2 PPPoE e Wireless

Nei capitoli precedenti sono stati illustrati gli strumenti: freeRADIUS, PPPoE e LDAP. Vedremo come questi strumenti possono essere utilizzati per offrire un controllo di accessi sufficientemente sicuro e decisamente flessibile e scalabile.

L'architettura di questo sistema prevede l'utilizzo di un NAS su cui è in esecuzione PPPoE-server. Questo server inoltrerà le richieste di autenticazione al server RADIUS, che a sua volta preleverà le informazioni degli utenti da una directory LDAP. In ambienti con un limitato numero di client, fino a 100, tutti e tre i servizi possono essere installati su una sola macchina.

Per dimensionare il NAS si può eseguire una approssimazione: ciascun cliente alloca uno spazio di memoria di 2 MB e richiede una "capacità di calcolo" di 2 MHz. Percui con 100 utenti basta e avanza un pentiun II a 400MHz con 256 MB di ram. Con un pentium III/1000 con 1GB di ram dovrebbe essere possibile installare tutto su un unica macchina: servizio PPPoE, server LDAP e freeRADIUS.

In ambienti più grandi si può prevedere di differenziare i ruoli su diverse macchine, come mostrato in figura 6.4, soluzione che necessità, però, di uno scambio di informazioni tra il RADIUS server e il server LDAP con connessione cifrata TLS.

6.2.1 Una soluzione concreta

Un sistema facilmente allestibile prevede la realizzazione di un NAS che offra tutti i servizi su un unica macchina opportunamente dimensionata.

PPPoE sarà configurato per ricevere richieste di autenticazione tramite il protocollo MSCHAPv2, e per cifrare la connessione con MPPE. I dati di autenticazione ricevuti saranno trasmessi, attraverso il radiusclient, al server freeRADIUS. Esso va configurato per accedere alla directory LDAP e prelevare l'hash *sambaNTPassword*, relativo all'utente da verificare, che sarà utilizzato per controllare le credenziali fornite.

Nella directory LDAP possono essere impostati, come già visto nel capitolo precedente, numerose informazioni di autenticazione tra cui il tempo massimo di connessione, l'indirizzo IP e il limite di traffico utilizzabile. Questo da la possibilità di considerare l'utilizzo della banda della rete come una risorsa su cui eventualmente applicare delle tariffe o dei limiti d'uso.

FreeRADIUS è configurato per salvare le informazioni di accounting su un database mysql, in questo modo è possibile monitorare le attività degli utenti, senza tuttavia analizzare il contenuto del traffico generato.

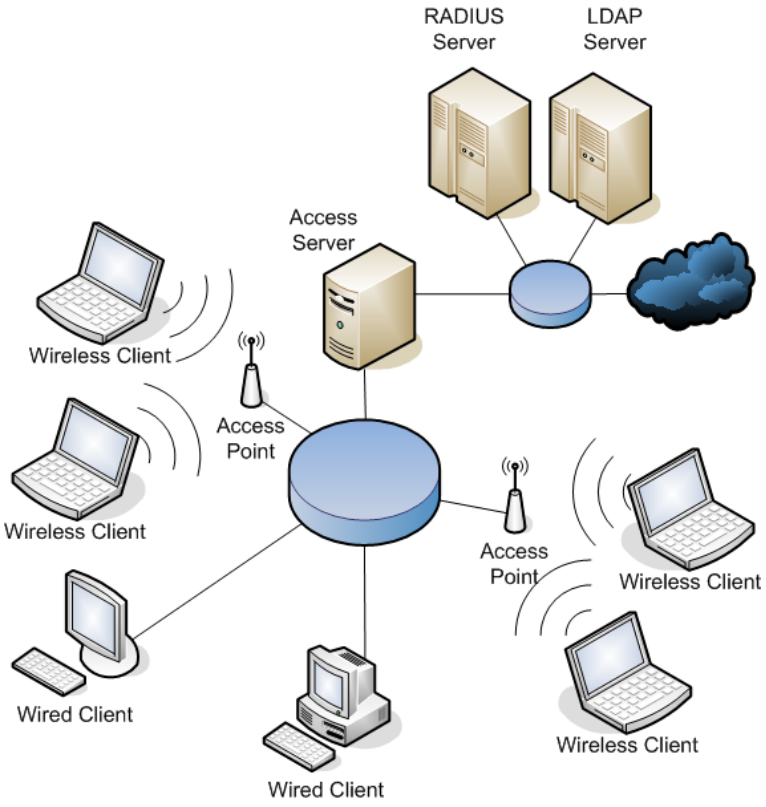


Figura 6.4: PPPoE e Reti Wireless

Questo sistema ha dei pregi:

- È compatibile con tutto l'harware attualmente in circolazione. Non è richiesto il supporto nè di WEP nè di WPA da parte degli access point.
- La configurazione dei client è immediata ed è la stessa sia per le connessioni via cavo che wireless.
- La soluzione funziona su tutti i principali sistemi operativi, anche nelle versioni non più recenti.
- Le informazioni di accesso sono centralizzate e utilizzabili per tutti gli altri sistemi di autenticazione.
- La gestione degli utenti è semplice e immediata.

- Sono disponibili informazioni sulle connessioni e i volumi di traffico di ciascun utente.

Ci sono tuttavia dei difetti, di cui già si è discusso:

- Il valore del MTU è inferiore a 1500 byte e comporta frammentazione dei pacchetti.
- Per utilizzare il protocollo MPPE è necessario ricompilare il kernel nei client linux, o disporne di uno recente.
- Sono note alcune vulnerabilità del protocollo MSCHAPV2.

Questa soluzione inizia a fare la sua comparsa anche in ambiete universitario. Le università di Trieste e di Trento offrono ai propri utenti un servizio analogo, ma meno completo. Infatti una delle due non usa connessioni cifrate (scelta molto discutibile), l'altra effettua l'autenticazione attraverso un meccanismo più contorto (passando per samba e quindi per LDAP) con prestazioni probabilmente più scadenti.

6.3 Wardriving

Il wardriving è una pratica discutibile e fuorilegge, nonchè fenomeno di moda, che vede sempre più persone che, equipaggiati con un computer portatile, con una scheda wireless munita di un'antenna con maggiore guadagno (e magari direzionale), gps e un veicolo, vanno alla ricerca di reti wireless potenzialmente insicure.

Esistono tools sempre più insidiosi, in grado di analizzare il traffico alla ricerca degli estremi per la connessione, altri capaci, come *airsnort* citato precedentemente, di eseguire la crittanalisi della chiave WEP. Esistono tool per eseguire anche attacchi DOS ad access point che usano il protocollo WPA. È persino disponibile una distribuzione linux chiamata *auditor* che raccoglie una incredibile quantità di tools: può essere paragonata alla cassetta degli attrezzi di uno scassinatore.

Difendersi dal Wardriving richiede degli accorgimenti, che tuttavia non possono essere sempre messi in atto:

- Impiegare access point con raggio d'azione limitato. Misurare l'intensità del segnale anche dall'esterno dell'edificio per valutarne le zone a maggiore rischio. Tuttavia i potenziali aggressori potrebbero avere antenne con una sensibilità maggiorata, percui potrebbero connettersi da posizioni più lontane a quelle che possiamo prevedere.
- Inibire l'access point a trasmettere in broadcast l'SSID. A costo di rendere più complessa la configurazione da parte degli utenti.
- Spegnere i dispositivi wireless nelle ore di chiusura degli uffici, soprattutto nelle ore notturne.

Figura 6.5: Auto attrezzata per il wardrive

Il wardriving è una delle principali insidie delle reti wireless, ma non è l'unica. Una breccia in una rete wireless potrebbe essere cercata anche per finalità di spionaggio (industriale e non) o per sfruttare la connessione per attività illegali. O semplicemente per risparmiare sulle spese di connessione.

6.4 Attacco a PPPoE

Immaginiamo un rompiscatole che riesca ad accedere alla nostra rete wireless. Questa procedura non è particolarmente complessa visto che i nostri access point sono configurati per trasmettere in broadcast il SSID e non

fanno alcun filtraggio sugli indirizzi MAC[1]. Inoltre non abbiamo attivato nè WEP nè WPA.

Una pacchia dirà il nostro "ospite". Ma lui ha bisogno di alcune informazioni fondamentali. Ha bisogno di un indirizzo IP da assegnare al suo host, ha la necessità di conoscere il gateway per instradare i suoi pacchetti; per ottenere queste informazioni probabilmente impiegherà uno sniffer.

Dopo aver sniffato qualche centinaio di pacchetti andrà ad analizzare cosa ha catturato, ma si sentirà spiazzato trovandosi solo frame ethernet contenenti pacchetti PPP: è solamente traffico di livello 2 (figura 6.6). Può darsi che tenterà di catturare ulteriore traffico, ma senza riuscire a capire molto di più: solo traffico PPP. Probabilmente a questo punto deciderà di cambiare bersaglio: puntare su una nuova rete wireless. Noi siamo salvi!

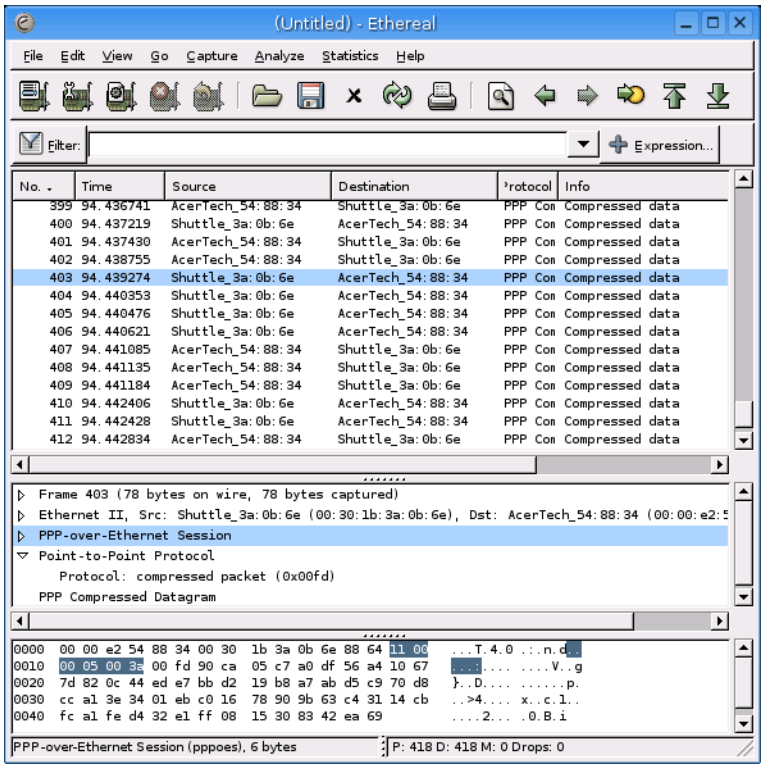


Figura 6.6: Sniff di una connessione PPPoE

Se invece il nostro tizio è più esperto ed ha più tempo a disposizione tenterà

[1]Soluzione inutile, visto che i MAC si possono cambiare facilmente

di ottenere un accesso dal nostro server PPPoE, cosa che un bravo sistemista noterà immediatamente dai file di log del server freeRADIUS. Tra l'altro è possibile impostare un numero massimo di tentativi a vuoto.

Certo, la sicurezza assoluta non esiste, ma questo sistema è sufficientemente valido per essere utilizzato in molte situazioni. Basti considerare che il PPPoE è il principale protocollo di autenticazione utilizzato dagli ISP.

6.5 Perchè no 802.1X?

Lo standard ieee 802.1X è un sistema di controllo degli accessi alle reti, locali e non, attraverso un sistema di autenticazione che sfrutta il protocollo *EAP* (Extensible Authentication Protocol). A differenza del PPPoE è la soluzione nata appositamente per risolvere il problema degli accessi in una rete, ma sono già stati messi in risalto i principali difetti:

- Poca disponibilità di harware compatibile.
- Costo dei supplicant.
- Impossibilità di impostare un indirizzo fisso per un client.

6.5.1 Funzionamento di 802.1X

Il meccanismo di autenticazione di 802.1X prevede che l'autenticazione del client avvenga direttamente sul dispositivo di rete cui il client si collega, tipicamente uno switch ethernet o un access point. Questo dispositivo riceve le informazioni di autenticazione, inviate con il protocollo EAP, da un particolare software in esecuzione sul client, il *supplicant*, e le trasmette al server RADIUS. Se il server RADIUS concencede l'autorizzazione il client può finalmente connettersi.

802.1X non prevede un particolare protocollo di crittografia dei dati, ma è solo un sistema per gestire l'accesso alle infrastrutture di rete. In ambito wireless può essere utilizzato congiuntamente al WEP o, meglio, al WPA per offrire maggiori garanzia di riservatezza nello scambio dei dati. Utilizzando,

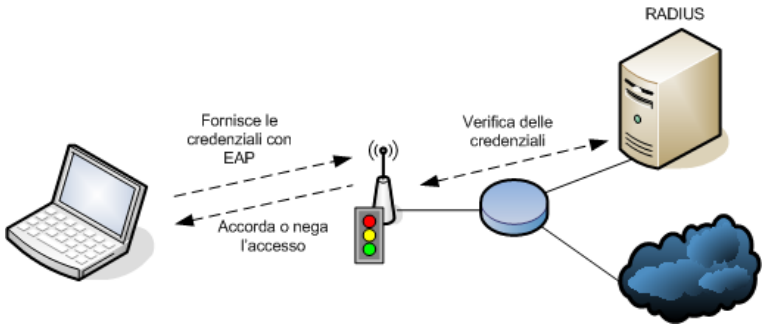


Figura 6.7: Autenticazione con 802.1X

per validare gli utenti, un server RADIUS consente di effettuare una valida gestione centralizzata degli accessi, ma non è possibile sfruttare alcune delle potenzialità disponibili con il PPPoE: assegnare indirizzi IP e ottenere diverse informazioni di accounting.

Il supplicant è l'applicativo lato client che si occupa di eseguire l'autenticazione 802.1X. È presente nativamente solo sulle versioni superiori a Microsoft Windows 2000, sui recenti MacOs e sulle piattaforme linux. Per le piattaforme sprovviste di supplicant è possibile acquistare dei supplicant commerciali.

Il problema più grande resta l'hardware. Nel 2004 allestii un sistema di autenticazione 802.1X in una rete wireless con alcune decine di utenti e quattro access point dislocati nell'edificio. Il sistema utilizzava WPA e un server RADIUS, l'autenticazione avveniva con EAP/TSL, con certificati preparati "in proprio" con openssl. Al momento di erogare il servizio mi sono reso conto che solo il 30% degli utenti aveva hardware compatibile con 802.1X e WPA, da cui la scelta sofferta di riprogettare interamente il servizio.

Quando 802.1X e WPA saranno supportati dal 90% dell'hardware a disposizione degli utenti, costituirà una valida alternativa al sistema PPPoE. Magari utilizzando una autenticazione EAP/PEAP e sfruttando nuovamente freeRADIUS e openLDAP per la gestione degli accessi.

6.5.2 EAP: Extensible Authentication Protocol

EAP, Extensible Authentication Protocol, è un framework di autenticazione universale normalmente utilizzato per connessioni wireless e point-to-point. EAP offre numerosi distinti metodi di autenticazione, di cui alcuni sono supportati dal protocollo WPA, e pertanto sfruttabili per il controllo degli accessi nelle reti wireless.

Valutiamo brevemente questi metodi:

EAP-MD5

Si tratta di una autenticazione incentrata su MD5, non molto sicuro in quanto esposto a possibili "dictionary attack". Tra l'altro, non verificando la rete su cui si connette, non protegge i client da eventuali access point abusivi impiegati per intercettare le informazioni di autenticazione.

EAP-TLS

È un metodo di autenticazione che sfrutta i certificati digitali X.509 trasferiti attraverso un *Transport Layer Security* (tsl). L'autenticazione è mutuale, evitando la connessione dei client a possibili dispositivi abusivi. Si tratta di uno dei più sicuri metodi di autenticazione, ma richiede una PKI.

EAP-LEAP

LEAP sta per Light EAP. Permette la mutua autenticazione tra client e dispositivi di accesso, e utilizza la coppia Username/Password per accordare l'accesso. A differenza di EAP-MD5 non è vulnerabile ad attacchi basati su dizionario ed è, nel complesso, un buon sistema di autenticazione.

EAP-TTSL

TTSL significa Tunneled Transport Layer Security. Deriva dal protocollo EAP-TLS, ma non richiede la presenza dei certificati per i client. L'autenticazione avviene in due fasi:

- Viene creato un tunnel sicuro tra client e server, utilizzando un algoritmo asimmettrico basato sulla chiave del server.

- Viene autenticato il client utilizzando un protocollo di autenticazione che può essere PAP, CHAP, MS-CHAP, MS-CHAPv2, ma anche EAP-MD5.

EAP-PEAP

Simile al EAP-TTSL, in quanto sfrutta un tunnel sicuro per l'invio delle informazioni, ma usa come protocollo di autenticazione EAP-MSCHAPv2

I metodi più interessanti sono EAP-TTSL e EAP-PEAP, che basano l'autenticazione sulla conoscenza della coppia username e password. EAP-TSL richiede l'acquisto di certificati X509 o la presenza di una PKI all'interno della struttura, soluzioni, entrambi, che comportano una spesa non trascurabile. Inoltre la certificazione aumenta la burocrazia e può richiedere una lunga attesa da parte degli utenti per ottenere l'accesso. Cosa che non è accettabile in realità dove ci sono ospiti che permangono in una struttura per poche ore, ma gli si vogliono comunque offrire dei servizi.

EAP-TTSL e EAP-PEAP possono essere utilizzati in un sistema implementato con un server RADIUS che va poi ad attingere i dati da un server LDAP. Questo consente una rapida e indolore migrazione dal sistema PPPoE al 802.1X.

6.6 Autenticazione EAP/PEAP con freeRADIUS e openLDAP

Attualmente PPPoE può essere una valida soluzione per implementare un sistema di controllo degli accessi sufficientemente solido. Probabilmente il futuro si orienterà verso una soluzione 802.1X o ulteriori evuoluzioni, quando si saranno superate le problematiche precedentemente elencate.

Il sogno di un sistemista è quello di non perdere intere giornate di lavoro per migrare da un sistema all'altro, ed in questo caso può essere accontentato: è possibile migrare da PPPoE a 802.1X senza dover reinserire tutti i dati degli utenti su un nuovo sistema di gestione. Infatti con piccole modifiche sul server freeRADIUS è possibile sfruttare la directory LDAP immutata per fini autenticativi.

La configurazione avviene, in grandi linee, in questo modo:

- Va generato (o acquistato) un certificato per il server. Si può utilizzare openssl per tale scopo.
- Bisogna configurare il modulo *tsl* nel file *eap.conf*. Un esempio può essere:

```
tls {
  private_key_password = supersecret
  private_key_file = ${raddbdir}/certs/cert-srv.pem
  CA_file = ${raddbdir}/certs/demoCA/cacert.pem
  dh_file = ${raddbdir}/certs/dh
  random_file = /dev/urandom
}
```

- Va configurato pure il modulo *peap*, sempre nel file *eap.conf*:

```
peap {
   default_eap_type = mschapv2
}
```

- Ancora sul file *eap.conf* bisogna impostare

```
default_eap_type = peap
```

- Nel file *radiusd.conf* bisogna aggiungere *eap* nelle sezioni *authorize* e *authenticate*

Configurando in modo appropriato i dispositivi di rete per sfruttare 802.1X e i supplicant presenti sui client sarà possibile accedere alla rete utilizzando sempre lo stesso username e la stessa password.

Capitolo 7

PPPoE e il quadro normativo

Negli ultimi tempi sono state emanate normative che vanno a discliplinare il settore informatico, mettendo dei paletti ben definiti su diversi aspetti di gestione di dati e utenti. Tale legislazione prevede pene consistenti/footnoteAnche fino a tre anni di reclusione per chi non la rispetta.

Il sistema implementato con PPPoE può essere un valido strumento per semplificare la gestione degli utenti e regolamentare opportunamente l'accesso ai dati e alle risorse informatiche.

7.1 La normativa sulla privacy

Il D. Lgs. 196/2003, e soprattutto l'allegato B, hanno introdotto l'obbligo da parte di enti e imprese di attuare delle misure minime di sicurezza, finalizzate a ridurre al minimo, mediante l'adozione di idonee e preventive misure di sicurezza, i rischi di distruzione o perdita, anche accidentale, dei dati stessi, di accesso non autorizzato o di trattamento non consentito o non conforme alle finalità della raccolta.

Il disciplinare tecnico prevede tra le misure minime[1] l'autenticazione informatica, adozione di procedure di gestione delle credenziali di autenticazione, l'utilizzazione di un sistema di autorizzazione; ma anche la protezione degli

[1] Art. 34: Trattamenti con strumenti elettronici

strumenti elettronici e l'utilizzo di tecniche di cifratura per la protezione dei dati sensibili.

Il sistema di autenticazione basato su pppoe + radius + ldap consente facilmente di utilizzare gli estremi di identificazione contenuti nella directory LDAP per tutti i servizi che richiedono il trattamento dei dati personali. Sempre seguendo la filosofia: "Una persona, uno username, una password".

L'autenticazione dei client con il protocollo MSCHAPv2 è indubbiamente abbastanza robusto, inoltre tutto il traffico generato tra il cliente e il pppoe server può essere cifrato con mppe, sia nel caso di connessione via cavo, sia per connessioni wireless In questo modo la comunicazione è sufficientemente protetto da orecchie indiscrete.

Un buon sistema per adeguarsi alla normativa della privacy può essere di configurare un NAS/footnoteNAS: Network Attached Storage server a cui ci si accede utilizzando il protocollo smb, in un dominio autenticato con samba + ldap. A quel punto gli utenti possono trasferire senza problemi i propri documenti sul NAS, su cui saranno effettuati regolari controlli con software antivirus[2] e i backup periodici.

7.2 La legge antiterrorismo

Recentemente è stata introdotto una normativa, familiarmente chiamato "decreto antiterrorismo"[3] che prevede che chi fornisce accessi a internet debba identificare preventivamente gli utenti. Questa legge piomba in un contesto in cui si erano iniziati a diffondere punti di accesso wireless presso stazioni, aereoporti e perfino supermercati; creando non poche problematiche per progettare una tipologia di servizi più complessa, con il rischio di dover buttare via tutto l'harware già acquistato.

La soluzione pppoe + radius + ldap può risolvere anche questo problema. Infatti basta che nell'atto della registrazione l'addetto faccia le opportune verifiche dell'identità dell'utente, facendo una copia del documento d'identità. L'inserimento dei dati nel server ldap possono essere agevolemente

[2] Esiste un antivirus opensource chiamato clamav

[3] D. lgs. 18/5/2005

fatti utilizzando, ad esempio, l'ambiente Ldap Account Manager. Magari utilizzando una connessione ssl verso il server aaa.

Le informazioni di accounting, ovvero data e ora di inizio e fine sessione e l'indirizzo ip assegnato possono essere memorizzato, come già detto, su una tabella mysql. Resta solamente da aggiungere delle opportune regole di firewalling per poter loggare anche il NAT, ricordando che tutte le connessioni verso l'esterno sono compiute usando l'indirizzo ip del server pppoe.

Una ulteriore soluzione potrebbe essere quella di impiegare un proxy server. Squid[4] offre un modulo per l'autenticazione con ldap. Inoltre è in fase di sviluppo un modulo per autenticarsi direttamente tramite un server radius.

[4]http://www.squid-cache.org

Appendice A

Il sistema PPPoE + freeradius + openldap

A.1 esempio: installazione server

Il server che ospiterà pppoe deve avere almeno 2 schede di rete. La potenza dell'elaboratore è direttamente proporzionale al numero di client ad esso connesso. Va poi considerato che ogni sessione occupa mediamente 2MB di memoria ram. Da alcume stime è possibile affermare che un server con processore Inter Pentium II a 400MHz con 512MB di ram possa gestire senza particolari problemi 250 client.

In questo esempio è stata impiegata la distribuzione linux OpenSUSE 10. Analoghi esperimenti positivi sono stati condotti con Debian GNU/Linux unstable (con kernel 2.6.15).

L'installazione del sistema è quella di default, la maggior parte della configurazione sarà poi fatta a mano.

Una volta conclusa l'installazione, verificare -usando yast2- la presenza dei seguenti pacchetti, provvedendo alla loro installazione qualora sia necessario:

```
kernel-default (2.6.13-15)
ppp (2.4.3-15)
rp-pppoe (3.5-338)
freeradius (1.0.4-4)
```

```
radiusclient (0.3.2-145)
openldap2 (2.2.27-6)
openlda2-client (2.2.27-6)
php4-ldap (4.4.0-6)
samba (3.0.20-4)
apache2 (2.0.54-10)
apache2-mod_php4 (4.4.0-6)
mysql (4.1.13-3)
mysql-client (4.1.13-3)
php4-mysql (4.4.0-6)
```

Il server deve essere configurato in modo che l'interfaccia eth0 sia connessa alla rete esterna, mentre la eth1 alla rete interna a cui accedono i nostri utenti. La eth0 sarà configurata utilizzando un indirizzo ip statico o dinamico a seconda delle diverse esigenze. L'interfaccia eth1 deve essere abilitata ma configurata, invece, senza nessun indirizzo ip.

Per fare questo bisogna trovare qual è il file di configurazione relativo alla scheda in questione nella directory /etc/sysconfig/network. Il file si chiama ifcfg-eth-id-[MAC ADDR] e va modificato, come nell'esempio che segue, lasciando vuoti i campi IPADDR, BROADCAST e NETWORK:

```
BOOTPROTO='static'
BROADCAST=''
IPADDR=''
MTU=''
NAME='Realtek RT8139'
NETWORK=''
REMOTE_IPADDR=''
STARTMODE='auto'
UNIQUE='JNkJ.IQxIdIhhuH7'
USERCONTROL='no'
_nm_name='bus-pci-0000:00:0b.0'
```

Per riconfigurare le schede di rete eseguire il comando:

```
# /etc/init.d/network restart
```

Usiamo il comando ifconfig per verificare lo stato di eth1:

```
# ifconfig eth1
eth1      Link encap:Ethernet  HWaddr 00:30:1B:3A:0B:6E
          inet6 addr: fe80::230:1bff:fe3a:b6e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3210 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4056 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:372089 (363.3 Kb)  TX bytes:4173375 (3.9 Mb)
          Interrupt:11 Base address:0x4000
```

Per abilitare il forward dalla eth1 alla eth0 editare il file /etc/sysconfig/sysctl e settare:

```
IP_FORWARD="yes"
```

Infine, per evitare sprechi di risorse utilizzare l'editor di runlevel presente su yast2 per ridurre allo stretto indispensabile in numero di daemons. Per evitare che venga avviata la sessione grafica è sufficiente editare il file /etc/inittab e modificare:

```
# The default runlevel is defined here
id:3:initdefault:
```

A.2 Installazione e configurazione di openldap

La configurazione di openldap è stata effettuata senza utilizzare tool grafici o script predefiniti.

Per aggiungere alcune features utili per l'uso con freeradius, aggiungere lo schema fornito con freeradius nella directory /etc/openldap/schema:

```
# cp /usr/share/doc/packages/freeradius/RADIUS-LDAPv3.schema \
 /etc/openldap/schema
```

Modificare il file /etc/openldap/slapd.conf come segue. In questo esempio è stato configurato per gestire la directory testdomain.it:

```
allow bind_v2
include         /etc/openldap/schema/core.schema
include         /etc/openldap/schema/cosine.schema
include         /etc/openldap/schema/nis.schema
include         /etc/openldap/schema/inetorgperson.schema
include         /etc/openldap/schema/samba.schema
include         /etc/openldap/schema/RADIUS-LDAPv3.schema
schemacheck     on
pidfile         /var/run/slapd/slapd.pid
argsfile        /var/run/slapd.args
loglevel        0
modulepath      /usr/lib/ldap
moduleload      back_bdb
backend         bdb
checkpoint 512 30
database        bdb
suffix          "dc=testdomain,dc=it"
directory       "/var/lib/ldap"
index           objectClass eq
lastmod         on
password-hash {CRYPT}
access to attrs=userPassword
        by dn="cn=admin,dc=testdomain,dc=it" write
        by anonymous auth
        by self write
        by * none
access to dn.base="" by * read
access to *
        by dn="cn=admin,dc=testdomain,dc=it" write
        by * read
```

Veniamo, quindi, alla creazione e alla popolazione della directory LDAP

Assicurarsi che slapd non sia in esecuzione:

```
# rcldap stop
```

Generare un hash per la password di administrator usando il comando slappasswd

```
# slappasswd -h {CRYPT} -s pippobaudo
{CRYPT}mAs60XkXq01Xo
```

Per inserire i primi dati su LDAP si può utilizzare un file organizzato come segue, e chiamato ldap.ldif. In questo esempio si crea l'utente admin del dominio testdomain.it utilizzando la password precedentemente generata.

```
dn: dc=testdomain,dc=it
dc: testdomain
o: testdomain.com
objectClass: top
objectClass: dcObject
objectClass: organization
structuralObjectClass: organization

dn: cn=admin,dc=testdomain,dc=it
cn: admin
userPassword: {CRYPT}mAs60XkXq01Xo
objectClass: top
objectClass: organizationalRole
objectClass: simpleSecurityObject
structuralObjectClass: organizationalRole
```

Per inserire i dati contenuti nel file ldap.ldif si può utilizzare il comando slapadd:

```
# slapadd -l ldap.ldif
```

A questo punto è possibile riavviare il demone ldap:

```
# rcldap start
```

Controllare il file /var/log/messages per controllare che non ci siano problemi:

```
# tail /var/log/messages

(...)
Mar 12 18:01:36 linux slapd[12973]: @(#) $OpenLDAP: slapd 2.2.27 \
(Sep  9 2005 17:48:51) \
abuild@d243:/usr/src/packages/BUILD/openldap-2.2.27/servers/slapd
```

Per popolare il database ldap è stato scelto un applicazione web-based chiamata Ldap Account Manager (http://lam.sourceforge.net/) che richiede un server web (in questo caso apache2, ma è da preferirsi apache-ssl), php4 e le librerie php4-ldap.

L'utilizzo di questo programma è intuitivo e ben documentato sul sito ufficiale.

Per accedere a ldap account manager è sufficiente utilizzare un browser e collegarsi all'indirizzo http://[ip del server aaa]/lam

Ldap account manager, una volta configurato, si occuperà da solo di creare le strutture per contenere gli utenti, i gruppi, gli host e i domini samba. In seguito si potrà procedere a inserire, nell'ordine, il dominio (o i domini) samba, i gruppi e quidi gli utenti.

A.3 Configurazione di freeradius

Freeradius è configurato per autenticare gli utenti usando l'autenticazione MSCHAPv2. I dati di accesso vengono prelevati dalla nostra directory ldap. La connessione tra freeradius e ldap avviene in chiaro, in quanto i servizi girano sulla stessa macchina. Nel caso, invece, in cui radius e ldap risiedono in host differenti è auspicabile l'uso del tsl.
Il file /etc/raddb/radiusd.conf è modificato come segue:

```
prefix = /usr
exec_prefix = /usr
sysconfdir = /etc
localstatedir = /var
sbindir = ${exec_prefix}/sbin
logdir = /var/log/radius
raddbdir = /etc/raddb
radacctdir = ${logdir}/radacct
confdir = ${raddbdir}
run_dir = ${localstatedir}/run/radiusd
log_file = ${logdir}/radiusd.log
libdir = /usr/lib/freeradius
pidfile = ${run_dir}/radiusd.pid
user = radiusd
group = radiusd
```

```
max_request_time = 30
delete_blocked_requests = no
cleanup_delay = 5
max_requests = 1024
bind_address = *
port = 0
hostname_lookups = no
allow_core_dumps = yes
regular_expressions     = yes
extended_expressions    = yes
log_stripped_names = no
log_auth = yes
log_auth_badpass = yes
log_auth_goodpass = yes
usercollide = no
lower_user = no
lower_pass = no
nospace_user = no
nospace_pass = no
checkrad = ${sbindir}/checkrad
security {
        max_attributes = 200
        reject_delay = 1
        status_server = no
}
proxy_requests  = no
$INCLUDE  ${confdir}/clients.conf
snmp    = no
thread pool {
        start_servers = 5
        max_servers = 32
        min_spare_servers = 3
        max_spare_servers = 10
        max_requests_per_server = 0
}
modules {
$INCLUDE ${confdir}/eap.conf
        mschap {
                authtype = MS-CHAP
                use_mppe = yes
                require_encryption = yes
                require_strong = yes
                with_ntdomain_hack = no
```

```
 }

ldap  {
          server = "localhost"
          identity = "cn=admin,dc=testdomain,dc=it"
          password = testpass
          basedn = "ou=people,dc=testdomain,dc=it"
          filter = "(uid=%{Stripped-User-Name:-%{User-Name}})"
          base_filter = "(objectclass=radiusprofile)"

         start_tls = no
          dictionary_mapping = ${raddbdir}/ldap.attrmap

          ldap_cache_timeout = 120
          ldap_cache_size = 0
          ldap_connections_number = 10

          password_attribute = sambaNTPassword
          timeout = 4
          timelimit = 3
          net_timeout = 1
          compare_check_items = no
 }
realm suffix {
        format = suffix
        delimiter = "@"
        ignore_default = no
        ignore_null = no
}
realm ntdomain {
        format = prefix
        delimiter = "\\"
        ignore_default = no
        ignore_null = no
}

preprocess {
        huntgroups = ${confdir}/huntgroups
        hints = ${confdir}/hints
        with_ascend_hack = no
        ascend_channels_per_line = 23
        with_ntdomain_hack = no
        with_specialix_jetstream_hack = no
```

```
        with_cisco_vsa_hack = no
}
files {
        usersfile = ${confdir}/users
        acctusersfile = ${confdir}/acct_users
        compat = no
}
detail {
        detailfile = ${radacctdir}/%{Client-IP-Address}/detail-%Y%m%d
        detailperm = 0600
}
detail auth_log {
         detailfile = ${radacctdir}/%{Client-IP-Address}/auth-detail-%Y%m%d
         detailperm = 0600
}
acct_unique {
        key = "User-Name, Acct-Session-Id, NAS-IP-Address,\
        Client-IP-Address, NAS-Port"
}
$INCLUDE  ${confdir}/sql.conf

radutmp {
        filename = ${logdir}/radutmp
        username = %{User-Name}
        case_sensitive = yes
        check_with_nas = yes
        perm = 0600
        callerid = "yes"
}
radutmp sradutmp {
        filename = ${logdir}/sradutmp
        perm = 0644
        callerid = "no"
}
attr_filter {
        attrsfile = ${confdir}/attrs
}
counter daily {
        filename = ${raddbdir}/db.daily
        key = User-Name
        count-attribute = Acct-Session-Time
        reset = daily
        counter-name = Daily-Session-Time
```

```
                check-name = Max-Daily-Session
                allowed-servicetype = Framed-User
                cache-size = 5000
        }
        always fail {
                rcode = fail
        }
        always reject {
                rcode = reject
        }
        always ok {
                rcode = ok
                simulcount = 0
                mpp = no
        }
        expr {
        }
        digest {
        }
        exec {
                wait = yes
                input_pairs = request
        }
        exec echo {
                wait = yes
                program = "/bin/echo %{User-Name}"
                input_pairs = request
                output_pairs = reply
        }
        ippool main_pool {
                range-start = 192.168.1.1
                range-stop = 192.168.3.254
                netmask = 255.255.255.0
                cache-size = 800
                session-db = ${raddbdir}/db.ippool
                ip-index = ${raddbdir}/db.ipindex
                override = no
                maximum-timeout = 0
        }
}
instantiate {
        exec
        expr
```

```
}
authorize {
         preprocess
         auth_log
         suffix
         files
         ldap
         mschap
}
authenticate{
          Auth-Type MS-CHAP {
                  mschap
          }
}
preacct {
        preprocess
        suffix
}
accounting {
        detail
        radutmp
}
session {
        radutmp
}
post-auth {
}
pre-proxy {
}
post-proxy {
}
```

Il file /etc/raddb/users deve contenere le righe:

```
DEFAULT Auth-Type = "MS-CHAP"
        Fall-Through = yes
```

A.4 Configurazione di rp_pppoe e pppd

I file di configurazione di rp_pppoe e pppd risiedono tutti nella directory /etc/ppp.

Il file pppoe-server-option contiene la configurazione di pppoe-server. Nel nostro caso contiene poche righe:

```
 refuse-pap
 refuse-chap
 require-mschap
 require-mschap-v2
 require-mppe
```

Più complessa è la configurazione di pppd, contenuta nel file options

```
name PPPoE-R
noipdefault
ipcp-accept-local
ipcp-accept-remote
debug
crtscts
lock
local
asyncmap 0
mru 1490
mtu 1490
nodetach
lcp-echo-interval 30
lcp-echo-failure 4
lcp-max-configure 60
lcp-restart 2
idle 600
noipx
file /etc/ppp/filters
noreplacedefaultroute
ms-dns 193.204.8.13
ms-dns 193.205.94.12
nobsdcomp
auth
refuse-pap
refuse-chap
require-mschap
require-mschap-v2
require-mppe
mppe-stateful
lcp-echo-interval 10
```

```
lcp-echo-failure 2
plugin radius.so
```

In questo caso vogliamo che la connessione avvenga utilizzando la cifratura MPPE a 128 bit e l'autenticazione con MSCHAP o MSCHAPv2. Tutte le altre richieste saranno rigettate.

A.5 Configurazione di radiusclient

Radiusclient si occupa di mettere in comunicazione il pppd con il server radius, utilizzando l'autenticazione MSCHAP o MSCHAPv2. Purtroppo nelle recenti distribuzioni manca il relativo dizionario ed è necessario crearlo ex-novo. Chiameremo questo dizionario dictionary.microsoft:

```
VENDOR          Microsoft       311     Microsoft
ATTRIBUTE       MS-CHAP-Response        1       string  Microsoft
ATTRIBUTE       MS-CHAP-Error           2       string  Microsoft
ATTRIBUTE       MS-CHAP-CPW-1           3       string  Microsoft
ATTRIBUTE       MS-CHAP-CPW-2           4       string  Microsoft
ATTRIBUTE       MS-CHAP-LM-Enc-PW       5       string  Microsoft
ATTRIBUTE       MS-CHAP-NT-Enc-PW       6       string  Microsoft
ATTRIBUTE       MS-MPPE-Encryption-Policy 7     string  Microsoft
ATTRIBUTE       MS-MPPE-Encryption-Type 8       string  Microsoft
ATTRIBUTE       MS-MPPE-Encryption-Types  8     string  Microsoft
ATTRIBUTE       MS-RAS-Vendor           9       integer Microsoft
ATTRIBUTE       MS-CHAP-Domain          10      string  Microsoft
ATTRIBUTE       MS-CHAP-Challenge       11      string  Microsoft
ATTRIBUTE       MS-CHAP-MPPE-Keys       12      string  Microsoft encrypt=1
ATTRIBUTE       MS-BAP-Usage            13      integer Microsoft
ATTRIBUTE       MS-Link-Utilization-Threshold 14 integer        Microsoft
ATTRIBUTE       MS-Link-Drop-Time-Limit 15      integer Microsoft
ATTRIBUTE       MS-MPPE-Send-Key        16      string  Microsoft
ATTRIBUTE       MS-MPPE-Recv-Key        17      string  Microsoft
ATTRIBUTE       MS-RAS-Version          18      string  Microsoft
ATTRIBUTE       MS-Old-ARAP-Password    19      string  Microsoft
ATTRIBUTE       MS-New-ARAP-Password    20      string  Microsoft
ATTRIBUTE       MS-ARAP-PW-Change-Reason 21     integer Microsoft
ATTRIBUTE       MS-Filter               22      string  Microsoft
ATTRIBUTE       MS-Acct-Auth-Type       23      integer Microsoft
ATTRIBUTE       MS-Acct-EAP-Type        24      integer Microsoft
```

```
ATTRIBUTE       MS-CHAP2-Response       25      string  Microsoft
ATTRIBUTE       MS-CHAP2-Success        26      string  Microsoft
ATTRIBUTE       MS-CHAP2-CPW            27      string  Microsoft
ATTRIBUTE       MS-Primary-DNS-Server   28      ipaddr
ATTRIBUTE       MS-Secondary-DNS-Server 29      ipaddr
ATTRIBUTE       MS-Primary-NBNS-Server  30      ipaddr
ATTRIBUTE       MS-Secondary-NBNS-Server 31     ipaddr
VALUE           MS-BAP-Usage            Not-Allowed     0
VALUE           MS-BAP-Usage            Allowed         1
VALUE           MS-BAP-Usage            Required        2
VALUE   MS-ARAP-PW-Change-Reason        Just-Change-Password            1
VALUE   MS-ARAP-PW-Change-Reason        Expired-Password                2
VALUE   MS-ARAP-PW-Change-Reason        Admin-Requires-Password-Change  3
VALUE   MS-ARAP-PW-Change-Reason        Password-Too-Short              4
VALUE           MS-Acct-Auth-Type       PAP             1
VALUE           MS-Acct-Auth-Type       CHAP            2
VALUE           MS-Acct-Auth-Type       MS-CHAP-1       3
VALUE           MS-Acct-Auth-Type       MS-CHAP-2       4
VALUE           MS-Acct-Auth-Type       EAP             5
VALUE           MS-Acct-EAP-Type        MD5             4
VALUE           MS-Acct-EAP-Type        OTP             5
VALUE           MS-Acct-EAP-Type        Generic-Token-Card      6
VALUE           MS-Acct-EAP-Type        TLS             13
END-VENDOR Microsoft
```

Per utilizzare questo nuovo dizionario è necessario aggiungere alla fine del file dictionary la seguente riga:

```
INCLUDE /etc/radiusclient/dictionary.microsoft
```

A questo punto è possibile procedere con la configurazione di radiusclient, editando il file radiuscliente.conf:

```
auth_order      radius,local
login_tries     4
login_timeout   60
nologin /etc/nologin
issue   /etc/radiusclient/issue
authserver      localhost
acctserver      localhost
servers         /etc/radiusclient/servers
dictionary      /etc/radiusclient/dictionary
```

```
login_radius    /usr/sbin/login.radius
seqfile         /var/run/radius.seq
mapfile         /etc/radiusclient/port-id-map
default_realm
radius_timeout  10
radius_retries  3
login_local     /bin/login
```

L'ultimo passo per la configurazione di radiusclient è quello di impostare le informazioni per la connessione al server radius, che sono contenute in /etc/radiusclient/servers:

```
#Server Name or Client/Server pair              Key
#----------------                               ---------------
#portmaster.elemental.net                       hardlyasecret
#portmaster2.elemental.net                      donttellanyone
#
# uncomment the following line for simple testing of radlogin
# with radiusd 1.16.1
localhost/localhost   pippobaudo
```

A.6 Avvio di PPPoE Server

Per avviare pppoe-server è stato scritto un file che, inoltre, abilita anche il NAT per consentire ai client connessi sulla rete interna di accedere alla rete esterna.

Il file, chiamato start-pppoe va poi inserito nelle rc.d per eseguirlo all'avvio del sistema:

```
#!/bin/bash
MAX=250
BASE=10.67.7.1
NAT=10.67.7.0/24
MYIP=193.205.94.13
/usr/sbin/iptables -A INPUT -i eth0 -s $NAT -j DROP
/usr/sbin/iptables -t nat -A POSTROUTING -s $NAT -j SNAT --to-source $MYIP
/usr/sbin/pppoe-server -T 60 -I eth1 -N $MAX -C PPPoE-R -S PPPoE-R -R $BASE
```

Restano da digitare alcuni comandi per far esegire lo script all'avvio:

```
# cp start-pppoe /etc/rc.d
# chmod 755 /etc/rc.d/start-pppoe
# chkconfig start-pppoe 35
```

A.7 Accounting con mysql

```
# mysqladmin create radius
# mysql radius </usr/share/doc/packages/freeradius/db_mysql.sql
```

Ricordarsi di aggiungere sql su /etc/raddb/radiusd.conf nella sezione accounting.

Installare phpmyadmin scaricandolo dal sito www.phpmyadmin.net

Scompattare il file e copiare la cartella phpMyAdmin-x.y.z in /usr/share/phpmyadmin e quindi cambiare i permessi.

```
# cd /var/www/html/phpmyadmin
# chown wwwrun:www -R /var/www/html/phpmyadmin
```

Copiare il file di configurazione di default, dovrebbe andare bene così

```
# cp libraries/config.default.php ./config.inc.php
```

Utilizzare phpmyadmin per settare una password all'utente root di mysql e configurare in modo sicuro gli accessi ai vari database.

Va creato anche un utente chiamato radius a cui consentire l'accesso al db radius.

Configurare il file /etc/raddb/sql.conf modificando i campi login (con radius) e password (la password settata)

Riavviare radius con il comando:

```
# rcradiusd restart
```

Nella tabella radact saranno registrati tutti gli accessi effettuati dai vari utenti, registrando indirizzo ip e durata della connessione.

Il risultato è molto gratificante, e parla da solo:

Tabella A.1: Contenuto della tabella radacct

UserName	StartTime	SessTime	IPAddress	InputOctets	OutputOctets
mario.rossi	2006-03-12 19:46:02	1392	10.67.7.4	1496892	67651364
mario.rossi	2006-03-12 20:09:48	951	10.67.7.6	2482	7534
rocco.demarco	2006-03-12 20:29:40 4	744	10.67.7.9	35004	1406274
rocco.demarco	2006-03-12 20:59:53	712	10.67.7.19	69978	2622140
rocco.demarco	2006-03-12 21:26:35	60	10.67.7.30	19668	743419

Appendice B

Analisi dei files di log

B.1 Log di una connessione pppoe con cifratura MPPE a 128 bit e autenticazione con MSCHAP

```
ppoe-server[7908]: Session 12 created for client 00:0d:88:99:0b:b1 (10.67.7.12)
 on eth1 using Service-Name ''
pppd[7908]: Plugin radius.so loaded.
pppd[7908]: RADIUS plugin initialized.
pppd[7908]: pppd 2.4.3 started by root, uid 0
pppd[7908]: using channel 12
pppd[7908]: Using interface ppp0
pppd[7908]: Connect: ppp0 <--> /dev/pts/1
pppd[7908]: sent [LCP ConfReq id=0x1 <mru 1490> <auth chap MS-v2>
<magic 0xcc571c7a>]
pppd[7908]: rcvd [LCP ConfAck id=0x1 <mru 1490> <auth chap MS-v2>
<magic 0xcc571c7a>]
pppd[7908]: rcvd [LCP ConfReq id=0x1 <magic 0x490a6ac3>]
pppd[7908]: sent [LCP ConfAck id=0x1 <magic 0x490a6ac3>]
pppd[7908]: sent [LCP EchoReq id=0x0 magic=0xcc571c7a]
pppd[7908]: sent [CHAP Challenge id=0x9a <091d9b20f796e13a2e500e09a2b4e3b3>,
name = "PPPoE-R"]
pppd[7908]: rcvd [LCP EchoReq id=0x0 magic=0x490a6ac3]
pppd[7908]: sent [LCP EchoRep id=0x0 magic=0xcc571c7a]
pppd[7908]: rcvd [LCP EchoRep id=0x0 magic=0x490a6ac3]
pppd[7908]: rcvd [CHAP Response id=0x9a
9b3a5335acb00000000000000000413e5766976e46d1cfae04091a9b747089b917124f8dd21800>,
name = "mario.rossi"]
```

```
pppd[7908]: sent [CHAP Success id=0x9a "S=1C32A1F91D1AF8AEFF35AA039816901AFCF3BE28"]
pppd[7908]: Script /etc/ppp/auth-up started (pid 7928)
pppd[7908]: sent [CCP ConfReq id=0x1 <mppe +H -M +S +L -D -C>]
pppd[7908]: Script /etc/ppp/auth-up finished (pid 7928), status = 0x0
pppd[7908]: rcvd [CCP ConfReq id=0x1 <mppe +H -M +S +L -D -C>]
pppd[7908]: sent [CCP ConfNak id=0x1 <mppe +H -M +S -L -D -C>]
pppd[7908]: rcvd [CCP ConfNak id=0x1 <mppe +H -M +S -L -D -C>]
pppd[7908]: sent [CCP ConfReq id=0x2 <mppe +H -M +S -L -D -C>]
pppd[7908]: rcvd [CCP ConfReq id=0x2 <mppe +H -M +S -L -D -C>]
pppd[7908]: sent [CCP ConfAck id=0x2 <mppe +H -M +S -L -D -C>]
pppd[7908]: rcvd [CCP ConfAck id=0x2 <mppe +H -M +S -L -D -C>]
pppd[7908]: MPPE 128-bit stateless compression enabled
pppd[7908]: sent [IPCP ConfReq id=0x1 <addr 10.0.0.1>]
pppd[7908]: rcvd [IPCP ConfReq id=0x1 <addr 0.0.0.0> <ms-dns1 0.0.0.0>
<ms-dns3 0.0.0.0>]
pppd[7908]: sent [IPCP ConfNak id=0x1 <addr 10.67.7.12> <ms-dns1 193.204.8.13>
<ms-dns3 193.205.94.12>]
pppd[7908]: rcvd [IPCP ConfAck id=0x1 <addr 10.0.0.1>]
pppd[7908]: rcvd [IPCP ConfReq id=0x2 <addr 10.67.7.12> <ms-dns1 193.204.8.13>
<ms-dns3 193.205.94.12>]
pppd[7908]: sent [IPCP ConfAck id=0x2 <addr 10.67.7.12> <ms-dns1 193.204.8.13>
<ms-dns3 193.205.94.12>]
pppd[7908]: local  IP address 10.0.0.1
pppd[7908]: remote IP address 10.67.7.12
pppd[7908]: Script /etc/ppp/ip-up started (pid 7934)
pppd[7908]: Script /etc/ppp/ip-up finished (pid 7934), status = 0x0
```

B.2 Log di una autenticazione su FreeRADIUS

```
rad_recv: Accounting-Request packet from host 127.0.0.1:1027, id=138, length=103
        Acct-Session-Id = "441464E7378F00"
        User-Name = "mario.rossi"
        Acct-Status-Type = Start
        Service-Type = Framed-User
        Framed-Protocol = PPP
        Acct-Authentic = RADIUS
        NAS-Port-Type = Virtual
        Framed-IP-Address = 10.67.7.15
        NAS-IP-Address = 192.168.33.1
        NAS-Port = 1
        Acct-Delay-Time = 0
  Processing the preacct section of radiusd.conf
```

```
modcall: entering group preacct for request 1
  modcall[preacct]: module "preprocess" returns noop for request 1
    rlm_realm: No '@' in User-Name = "mario.rossi", looking up realm NULL
    rlm_realm: No such realm "NULL"
  modcall[preacct]: module "suffix" returns noop for request 1
modcall: group preacct returns noop for request 1
  Processing the accounting section of radiusd.conf
modcall: entering group accounting for request 1
radius_xlat:  '/var/log/radius/radacct/127.0.0.1/detail-20060312'
rlm_detail: /var/log/radius/radacct/%{Client-IP-Address}/detail-%Y%m%d
expands to /var/log/radius/radacct/127.0.0.1/detail-20060312
  modcall[accounting]: module "detail" returns ok for request 1
radius_xlat:  '/var/log/radius/radutmp'
radius_xlat:  'mario.rossi'
  modcall[accounting]: module "radutmp" returns ok for request 1
modcall: group accounting returns ok for request 1
Sending Accounting-Response of id 138 to 127.0.0.1:1027
Finished request 1
Going to the next request
Waking up in 6 seconds...
```

B.3 Test di autenticazione con MSCHAP

Per testare il funzionamento dell'autenticazione MSCHAP direttamente sul radius server è possibile utilizzare il comando radauth fornito nella versione trial del software AXL RADIUS Client API:

```
# radauth.sh mario.rossi testpass MSCHAP2 ipradiuserver 1 passwordradius

Radtest running RADIUS client version 3.42 Non-Random Demonstration Version

------------------------------- Authentication -------------------------------
Authenticating: mario.rossi testpass
Sending to server 193.205.94.13:1812
Sending Attributes:
NAS-IP-Address (4), Length: 6, Data: [??^3], [# 3251461683] / [IP 193.205.94.51],
0xC1CD5E33
NAS-Port (5), Length: 6, Data: [# 1], 0x00000001


<9> ------------------- Request Packet -----------------
<9> Address: 193.205.94.13:1812  Packet Length: 127 Type: Access-Request(1)
0109007F0A0B0C0D - 0E0F101112131415  ........ - ........
```

```
161718190406C1CD - 5E33050600000001  ........ - ^3......
1A18000001370B12 - 2A2B2C2D2E2F3031  .....7.. - *+,-./01
3233343536373839 - 1A3A000001371934  23456789 - .:...7.4
1A001A1B1C1D1E1F - 2021222324252627  ........ -  !"#$%&'
2829000000000000 - 000086E98565B67C  ()...... - .....e.|
9B0CF70525B6C402 - 651953D8749A2845  ....%... - e.S.t.(E
3E89010D6D617269 - 6F2E726F737369__  >...mari - o.rossi

Attributes:
NAS-IP-Address (4), Length: 6, Data: [ï¿½ï¿½^3], [# 3251461683] / [IP 193.205.94.51],
0xC1CD5E33
NAS-Port (5), Length: 6, Data: [# 1], 0x00000001
Vendor-Specific ID: Microsoft (311), VSA Count: 1
   MS-CHAP-Challenge (11), Length: 18, Data: [*+,-./0123456789],
0x2A2B2C2D2E2F30313233343536373839
Vendor-Specific ID: Microsoft (311), VSA Count: 1
   MS-CHAP2-Response (25), Length: 52, Data:
0x1A001A1B1C1D1E1F202122232425262728290000000000000000086E98565B67C9B051953D8749A28453E89
User-Name (1), Length: 13, Data: [mario.rossi], 0x6D6172696F2E726F737369

<9> ------------------------------------------------

<9> ------------------- Response Packet -----------------
<9> Address: 193.205.94.13:1812  Packet Length: 179 Type: Access-Accept(2)
020900B3C9DEC8A8 - 4049FC031A538D1B  ........ - @I...S..
816692551A330000 - 01371A2D1A533D31  .f.U.3.. - .7.-.S=1
4342394531413546 - 3539423239304539  CB9E1A5F - 59B290E9
3732323235343333 - 3434374341373431  72225433 - 447CA741
323930344531371A - 2A00000137112490  2904E17. - *...7.$.
0D564F3868663FCF - 5A8448D1AD46C3C0  .VO8hf?. - Z.H..F..
F8DBCDCDFD37B18A - 37C3AF269D93568E  .....7.. - 7..&..V.
F11A2A0000013710 - 249DAECBBA904699  ..*...7. - $.....F.
AB149A0C42BF2C0A - CD69B8A7D61AE2DA  ....B.,. - .i......
64BC4D643EDC865E - 10F55A1A0C000001  d.Md>..^ - ..Z.....
3707060000000021A - 0C00000137080600  7....... - ....7...
000004__________ - _________________  ...

Attributes:
Vendor-Specific ID: Microsoft (311), VSA Count: 1
   MS-CHAP2-Success (26), Length: 45, Data:
 0x1A533D3143423945314135463539423239304539373232323534333334343743413734313239303445313 7
Vendor-Specific ID: Microsoft (311), VSA Count: 1
   MS-MPPE-Recv-Key (17), Length: 36, Data:
 0x900D564F3868663FCF5A8448D1AD46C3C0F8DBCDCDFD37B18A37C3AF269D93568EF1
Vendor-Specific ID: Microsoft (311), VSA Count: 1
   MS-MPPE-Send-Key (16), Length: 36, Data:
 0x9DAECBBA904699AB149A0C42BF2C0ACD69B8A7D61AE2DA64BC4D643EDC865E10F55A
Vendor-Specific ID: Microsoft (311), VSA Count: 1
   MS-MPPE-Encryption-Policy (7), Length: 6, Data: [# 2 (SLIP)], 0x00000002
Vendor-Specific ID: Microsoft (311), VSA Count: 1
```

```
   MS-MPPE-Encryption-Types (8), Length: 6, Data: [# 4], 0x00000004

<9> ------------------------------------------------
Authenticated
Attributes returned from server:
Vendor-Specific ID: Microsoft (311), VSA Count: 1
   MS-CHAP2-Success (26), Length: 45, Data:
 0x1A533D3143423945314135463539423239304539373232323534333334343743413734313239303445313 7
Vendor-Specific ID: Microsoft (311), VSA Count: 1
   MS-MPPE-Recv-Key (17), Length: 36, Data:
 0x900D564F3868663FCF5A8448D1AD46C3C0F8DBCDCDFD37B18A37C3AF269D93568EF1
Vendor-Specific ID: Microsoft (311), VSA Count: 1
   MS-MPPE-Send-Key (16), Length: 36, Data:
 0x9DAECBBA904699AB149A0C42BF2C0ACD69B8A7D61AE2DA64BC4D643EDC865E10F55A
Vendor-Specific ID: Microsoft (311), VSA Count: 1
   MS-MPPE-Encryption-Policy (7), Length: 6, Data: [# 2 (SLIP)], 0x00000002
Vendor-Specific ID: Microsoft (311), VSA Count: 1
   MS-MPPE-Encryption-Types (8), Length: 6, Data: [# 4], 0x00000004
```

Appendice C

Elenco RFC

Per comodità del lettore sono elencate le principali RFC inerenti alle tematiche affrontate in queste pagine:

C.1 AAA, RADIUS

RFC 2194 Review of Roaming Implementations
RFC 2477 Criteria for Evaluating Roaming Protocols
RFC 2881 Network Access Server Requirements Next Generation (NASREQNG) NAS Model
RFC 2903 Generic AAA Architecture
RFC 2904 AAA Authorization Framework
RFC 2905 AAA Authorization Application Examples
RFC 2906 AAA Authorization Requirements
RFC 3169 Criteria for Evaluating Network Access Server Protocols
RFC 3539 AAA Transport Profile
RFC 2865 Remote Authentication Dial In User Service (RADIUS)
RFC 2866 RADIUS Accounting
RFC 2548 Microsoft Vendor-specific RADIUS Attributes
RFC 2607 Proxy Chaining and Policy Implementation in Roaming
RFC 2618 RADIUS Authentication Client MIB
RFC 2619 RADIUS Authentication Server MIB
RFC 2620 RADIUS Accounting Client MIB

RFC 2621 RADIUS Accounting Server MIB
RFC 2809 Implementation of L2TP Compulsory Tunneling via RADIUS
RFC 2867 RADIUS Accounting Modifications for Tunnel Protocol Support
RFC 2868 RADIUS Attributes for Tunnel Protocol Support
RFC 2869 RADIUS Extensions
RFC 2882 Network Access Servers Requirements: Extended RADIUS Practices
RFC 3162 RADIUS and IPv6
RFC 3575 IANA Considerations for RADIUS
RFC 3576 Dynamic Authorization Extensions to RADIUS
RFC 3579 RADIUS Support for EAP
RFC 3580 IEEE 802.1X RADIUS Usage Guidelines
RFC 4014 RADIUS Attributes Suboption for the DHCP Relay Agent Information Option

C.2 PPP, PPPoE

RFC 2687, Proposed Standard, PPP in a Real-time Oriented HDLC-like Framing
RFC 2153, Informational, PPP Vendor Extensions
RFC 1994, PPP Challenge Handshake Authentication Protocol (CHAP)
RFC 1662, Standard 51, PPP in HDLC-like Framing
RFC 1661, Standard 51, The Point-to-Point Protocol (PPP)
RFC 2516 - A Method for Transmitting PPP Over Ethernet (PPPoE)
RFC 3817 - Layer 2 Tunneling Protocol (L2TP) Active Discovery Relay for PPP over Ethernet (PPPoE)
RFC 3078 - Microsoft Point-To-Point Encryption (MPPE) Protocol

C.3 LDAP

RFC 1487 - LDAPv2: The initial specification
RFC 2251 - LDAPv3: The specification of the LDAP on-the-wire protocol
RFC 2252 - LDAPv3: Attribute Syntax Definitions
RFC 2253 - LDAPv3: UTF-8 String Representation of Distinguished Names

RFC 2254 - LDAPv3: The String Representation of LDAP Search Filters
RFC 2255 - LDAPv3: The LDAP URL Format
RFC 2256 - LDAPv3: A Summary of the X.500(96) User Schema for use with LDAPv3
RFC 2829 - LDAPv3: Authentication Methods for LDAP
RFC 2830 - LDAPv3: Extension for Transport Layer Security
RFC 3377 - LDAPv3: Technical Specification
RFC 1823 - LDAP API (in C)
RFC 2247 - Use of DNS domains in distinguished names
RFC 2307 - Using LDAP as a Network Information Service
RFC 2798 - inetOrgPerson LDAP Object Class
RFC 2849 - The LDAP Data Interchange Format (LDIF)
RFC 2891 - Server Side Sorting of Search Results
RFC 3062 - LDAP Password Modify Extended Operation
RFC 3296 - Named Subordinate References in LDAP Directories
RFC 3383 - Internet Assigned Numbers Authority (IANA) Considerations for the Lightweight Directory Access Protocol (LDAP)
RFC 3866 - Language Tags and Ranges in LDAP
RFC 3909 - LDAP Cancel Operation

Bibliografia

[1] Jonathan Hassell, *Radius*, O'Reilly, 2002

[2] Tony Bautts, Terry Dawson, Gregor N. Purdy, *LINUX - Guida per l'amministratore di rete*, HOPS - tecniche nuove, 2005

[3] David F. Skoll, *PPPoE and Linux*, http://www.roaringpenguin.com

[4] David F. Skoll, *A PPPoE Implementation for Linux*, http://www.roaringpenguin.com

[5] Network Working Group, *A Method for Transmitting PPP over ethernet (PPPoe)*, rfc2516, 1999

[6] Corvin Light-William, Joshua Drake, *Linux PPP howto*, http://www.ildp.org, 2000

[7] G. Paternò, *Using PPPoE to authenticate Wireless Lans*, 2002

[8] G. Paternò, *Wireless Security: A scalable solution for consumer, corporations, ISP and mobile operators*

[9] G. Paternò, *Sicurezza nelle wireless LAN*, 2003

[10] Bruce Schneier, *Cryptanalysis of Microsoft Point-to-Point Tunneling Protocol (PPTP)*

[11] Roel Van Meer, Giuseppe Lo Biondo, *LDAP implementation HOWTO*, http://www.faqs.org ,2001

[12] Ken Roser, *EAP/TLS Setup for FreeRADIUS adn Windows XP supplicant*, 2002

[13] Stefano Tegliaferri, *Squid Book, oltre le FAQ*, 2005

[14] A. S. Tanenbaum, *Reti di Calcolatori*, Prentice Hall International

[15] S. Steinke, *Network Tutorial*, CMP book, 5^{th} edition

[16] P. Loshin, *TCP/IP*, AP Professional, 2^{nd} edition

[17] C. Pfleeger, S. Pfleeger, *Sicurezza in informatica*, Pearson/Prentice Hall

[18] D. W. Chadwick *Understanding X.500 - The Directory*, http://sec.cs.kent.ac.uk/x500book/

[19] D. Giacomini *Appunti di informatica libera*, http://a2.swlibero.org

Molte informazioni sono state attinte dalle usenet, in particolare sui gruppi *comp.protocol.ppp*, *comp.os.linux.networking*, *comp.os.linux.security*, *it.comp.os.linux.sys* e *it.comp.reti.locali*

Elenco delle figure

Elenco delle tabelle

Indice analitico