RETI E SICUREZZA

PROTOCOLLO RADIUS

Marilena Mordenti

27 GENNAIO 2006

INTRODUZIONE

Cosa succede quando un utente deve provare la propria identità ad un computer? Il computer deve essere dotato di un set di processi e protocolli per verificare che l'utente sia veramente chi dichiara di essere, fornire il servizio a cui l'utente può accedere quindi comunicarglielo.
C'è un protocollo che fa tutto questo: Remote Access Dialin User Service: RADIUS. Creato originariamente dalla Livinstone Enterprises, è un access-control protocol che verifica e autentica gli utenti usando il metodo challenge-response.

Per challenge-response si intende un sistema di autenticazione nel quale un partecipante genera un numero casuale, lo cifra e lo spedisce all'altra parte. L'altro partecipante decifra e rispedisce il risultato. Se quest'ultimo e' corretto prova al primo che il secondo conosce il segreto appropriato, necessario alla decifrazione. Esistono delle variazioni al sistema che utilizzano chiavi pubbliche o crittografia simmetrica. Alcune permettono un' autenticazione a due vie, in grado di assicurare a ciascun partecipante l'identita' dell'altro. Un sistema challenge-response non spedisce mai una password, ne' in chiaro ne' cifrata. E' inutile registrare i messaggi per riutilizzarli in seguito, il numero casuale e' differente per ogni sessione.

Radius viene utilizzato nei servizi gestiti dagli ISP legati a internet e comunque in ogni applicazione in cui occorre un'autenticazione centralizzata, un'autorizzazione regolata e un dettagliato accounting per gli utenti.

1 - IL MODELLO AAA

L'ambiente di lavoro, attorno al quale Radius è nato, è conosciuto come processo AAA: Autenticazione, Autorizzazione e Accounting.
Questo modello gestisce e tiene traccia di ogni transazione dall'inizio alla fine. Le principali domande alle quali intende rispondere sono: chi sei? Che servizi sei autorizzato ad utilizzare? Quali sono le tue attività?
L'architettura AAA è sicuramente una delle migliori metodologie di autenticazione utenti. Il maggiore problema dei sistemi precedenti è la scalabilità: mentre gestire gli accessi agli utenti in una piccola rete è relativamente semplice, se la rete deve connettersi ad altre, con diversi metodi di autenticazione, la gestione diventa molto complessa.
Il working group AAA fu formato attraverso l'IETF (Internet Engineering Task Force) per far fronte alle limitazioni derivanti dalla necessità di gestire l'autenticazione su reti eterogenee.

1.1 - CONTROLLO ACCESSO UTENTI

Il modello AAA focalizza la propria attenzione su tre aspetti del controllo di accesso utenti: autenticazione, autorizzazione e accounting.

- Autenticazione è il processo di verifica dell'identità dichiarata dalla persona o dalla macchina. Le più normali forme di autenticazione sono date dalla combinazione di ID (Identifier) e password. Il fatto che l'utente conosce la password dovrebbe bastare come prova dell'autorizzazione all'accesso. Tuttavia la distribuzione delle password rischia di

distruggere questo metodo di autenticazione. L'autenticazione si basa sul concetto di fiducia che si crea tra il server e il client. AAA deve essere implementato per reti eterogenee e supportare diversi tipi di client e servizi.

- Autorizzazione consiste nella considerazione sulle attività e sulle impostazioni che devono essere fornite ad un utente autenticato. Ad esempio, riguardo a un ISP (Internet Service Provider), questo può decidere di fornire un ip statico, oppure un ip dinamico assegnato attraverso un servizio di DHCP (Dynamic Host Configuration Protocol).
- Accounting include quelle che sono le regole relative alla connessione quali ad esempio, la durata o i dati che l'utente, può ricevere o spedire, nell'ambito della sua sessione.

Il punto fondamentale è che l'architettura AAA, dovendo lavorare in situazioni di eterogeneità di utenti e architetture di rete, deve essere il più neutrale possibile.

Il modello AAA dipende da un'interazione client-server, dove il client chiede servizi e risorse al server. I servers, che possono essere più di uno, possono essere distribuiti e decentrati sulla rete. I client che richiedono servizi e risorse, possono ottenere direttamente risposta dal server che interrogano, oppure il server AAA può fare da proxy e passare la richiesta ad un altro server AAA.

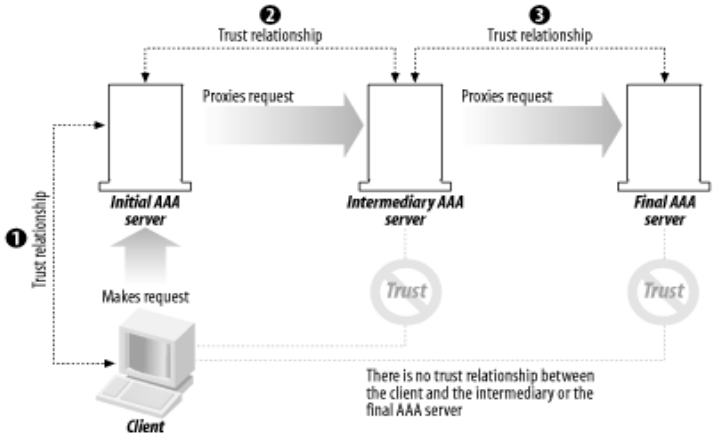
1.2 – MODELLI

Ci sono due diversi modelli definiti nell'architettura AAA:

- Modello Hop-to-hop:

Qui il client fa la sua richiesta al primo server (Initial AAA server) con il quale c'è una relazione di fiducia. Il server fa da proxy e contatta il server intermedio, con il quale c'è a sua volta una relazione di fiducia e via di seguito fino al server AAA finale. Non esiste una relazione di fiducia tra il client e il server AAA finale: la fiducia si instaura ad ogni salto.

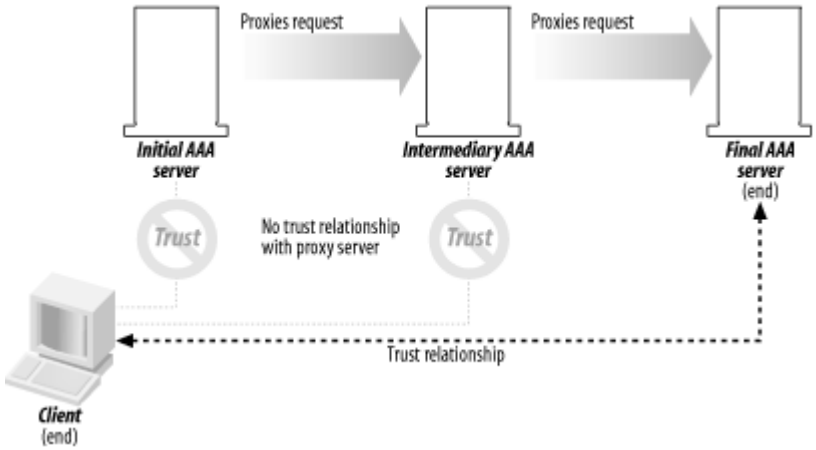
Indipendenza di relazioni in transazione hop-to-hop:



- Modello End-to-end:

Qui la relazione di fiducia è solo tra il client e il server AAA finale. Non esistendo relazioni di trusting tra i servers, sono necessari metodi di protezione e validazione dell'integrità dei dati, durante il percorso. In queste situazioni si usano comunemente certificati digitali o altri certificati PKI (Public Key Infrastructure).

Client/server: relazione di fiducia nel modello end-to-end.

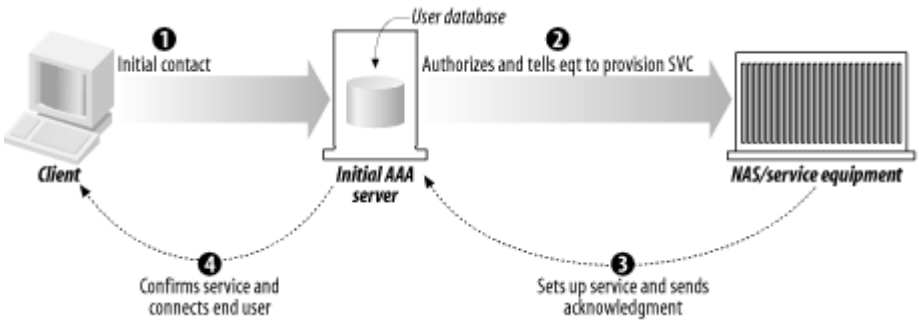


1.3 - SEQUENZE DI AUTORIZZAZIONE

Si parla di UHO (User Home Organization) in riferimento all'entità che ha il contratto direttamente con l'utente finale e dei Service Provider (SP) che forniscono e gestiscono gli apparati tangibili di rete. Non necessariamente l'UHO e l'SP sono la stessa organizzazione.
Nel caso in cui l'UHO e il SP siano della stessa organizzazione: ci sono diversi metodi attraverso i quali l'utente finale, il server AAA e il network equipment interagiscono durante la transazione.
Nel dettaglio vediamo tre diverse possibili sequenze:

- Agent sequence: il server AAA agisce come "middleman", cioè è tra il client e il service equipment.

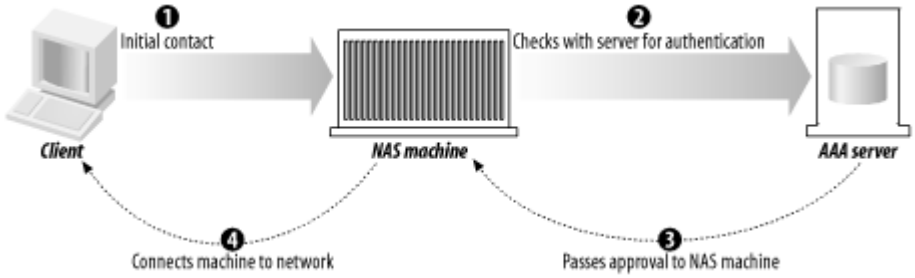
Agent sequence



SVC (Switched Virtual Circuit ovvero a commutazione di pacchetto virtuale - Connessioni paragonabili ad una connessione telefonica: si stabilisce una connessione, si trasferiscono i dati ed infine la connessione viene chiusa)

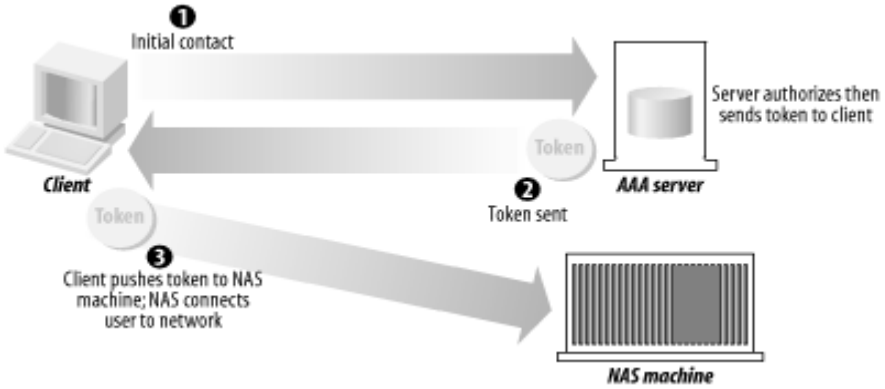
- Pull sequence: qui l'utente contatta direttamente il service equipment che a sua volta "prende" l'autorizzazione dal server AAA.

Pull sequence



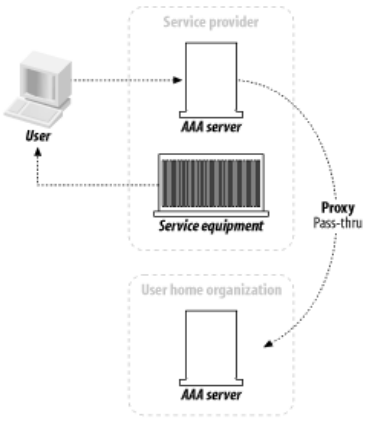
- Push sequence: il client riceve l'autorizzazione dal server AAA e lo passa al service equipment.

Push sequence

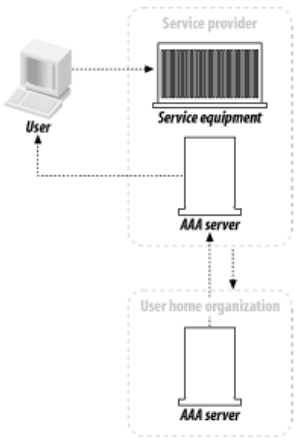


Nelle possibili sequenze di autorizzazione indicate sopra, il service equipment e il server AAA sono sotto controllo e di proprietà di un singolo UHO. Tuttavia, se il service equipment è di proprietà ed è gestito da un'altra organizzazione, si parla di roaming. E' una realtà molto diffusa: un utente si connette a un set di accessi dial-up che l'ISP noleggia da un service provider più grande, il service equipment è su un altro dominio. Le sequenze di autorizzazioni agent, push e pull, come si vede nelle immagini sotto riportate, sono possibili anche quando siamo in presenza di roaming.

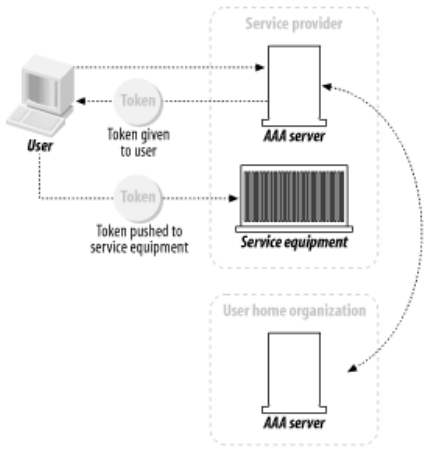
The roaming agent sequence:



The roaming pull sequence:

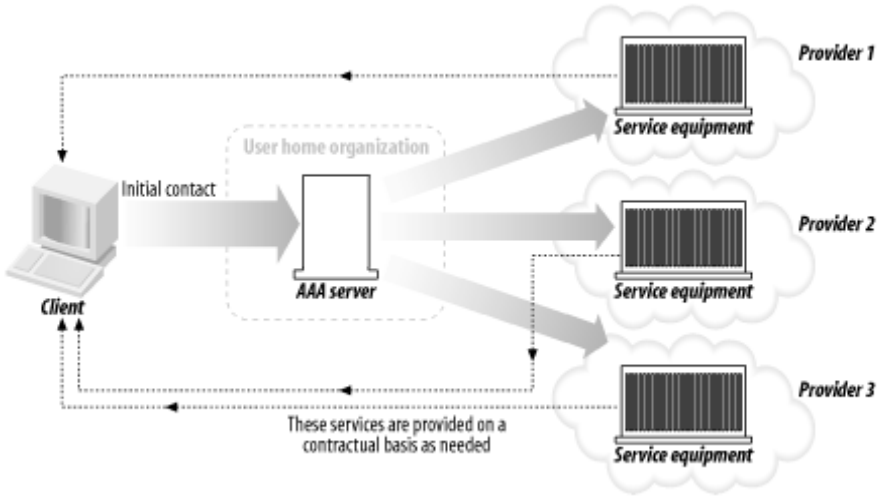


The roaming push sequence:



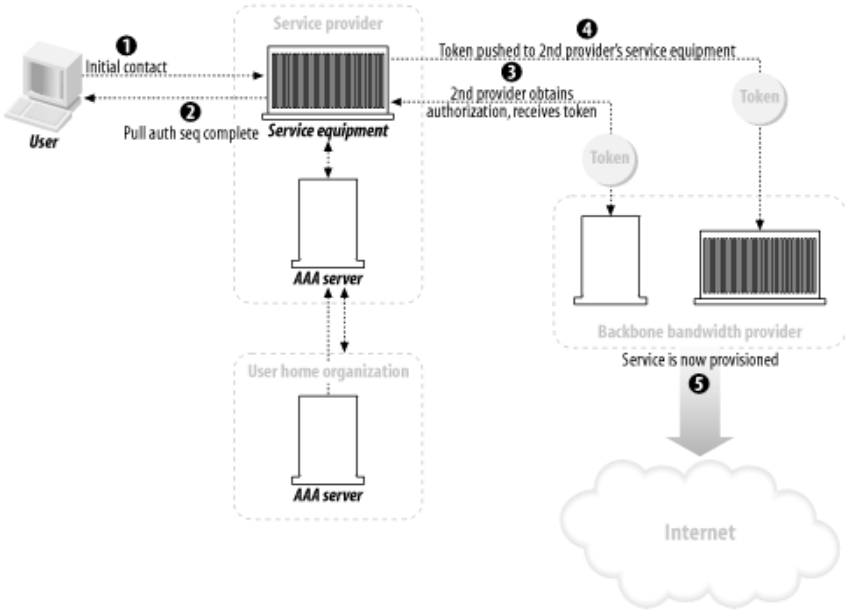
Parliamo di servizio distribuito nel caso in cui un service provider fa contratti con molti rivenditori, che a loro volta, forniscono il servizio a molti clienti. Ad esempio un provider può garantire una certa banda per una particolare azienda. L'ISP di frontline con il quale il rivenditore, come cliente, contratta, occorre che garantisca QoS sulla rete al fine di mantenere l'affidabilità. Il cliente in questo caso usa un servizio distribuito:

Modello di servizio distribuito



Nella figura successiva vediamo come un servizio distribuito può coinvolgere diversi SP. Qui l'utente può contattare il service equipment nel 1^salto usando la sequenza "pull uthorization". Dopodiché l'ISP equipment può usare una sequenza di "push": contatta il server AAA della 2^ organizzazione e ottiene l'autorizzazione.

Push and pull authorization sequences per fornire ad un utente un servizio distribuito



I servizi distribuiti e il roaming possono creare nuovi modelli infrastrutturali: un'azienda può avere come unico scopo di esistenza l'attività di autorizzazione e autenticazione per una varietà di reti diverse.

1.4 - POLICIES

Si tratta delle regole nella fase in cui i server AAA devono disporre di informazioni che gli permettono di verificare se le richieste degli utenti sono accettabili. Queste informazioni possono essere immagazzinate nei modi più svariati: data base o semplici files di testo, ed essere anche distribute. La cosa importante è che queste regole siano reperibili, valutabili, applicabili e comunque consentire interrogazioni attraverso protocolli come LDAP.

2 - RADIUS

Radius nasce dalla necessità di avere un metodo di autenticazione, autorizzazione e accounting per utenti che necessitano di accedere a servizi da macchine eterogenee.

2.1 - CARATTERISTICHE GENERALI E LIMITI

Gli RFCs (Request for Comments) specificano le caratteristiche del protocollo Radius.

- è basato su UDP (User Datagram Protocol)
- è connectionless (le problematiche inerenti la mancata ricezione dei dati vengono gestiti a livello superiore) e usa connessioni dirette
- utilizza il modello hop-by-hop
- è stateless
- supporta PAP e CHAP via PPP.

  (PAP = Password Authentication Protocol – il metodo utilizzato dal protocollo PAP è di tipo 2-way handshake: quando la connessione è stabilita viene inviata da parte del client una coppia Username e Password; il server confronta i dati ricevuti con quelli nel proprio database quindi autentica o rifiuta il client.)

  (CHAP = Challenge Handshake Authentication Protocol - utilizza un metodo 3-way handshake per verificare l'identità del client; questo processo può essere ripetuto più volte lungo la durata della connessione; una volta stabilito il collegamento il server invia una stringa contenente dei caratteri ed il proprio nome, chiamata "challenge"; il client invia in risposta una stringa calcolata tramite una funzione hash one-way: il server controlla che l'hash che ha generato anch'esso tramite la stessa funzione one-way e, se è corretto, autentica il client altrimenti termina la connessione.

  PAP non è un metodo di autenticazione molto robusto in quanto l'autenticazione avviene inviando la password in chiaro da parte del client. Altro punto debole di PAP è dovuto all'utilizzo di password riutilizzabili. CHAP utilizza invece un metodo più sicuro in quanto non invia la password sul collegamento ed il challange utilizzato per l'autenticazione, oltre ad essere crittografato tramite MD5, è diverso ogni volta.

  PPP = Point to Point Protocol – protocollo di livello 2 che fornisce un metodo standard per il trasporto di diversi tipi di protocollo su una connessione punto a punto; sviluppato per facilitare il collegamento tra apparati di rete eterogenei.)
- usa MD5 come algoritmo di cifratura (MD5 = Message Digest 5, un algoritmo in grado di associare a dei dati qualsiasi, una stringa numerica di controllo a 128bit sempre uguale a parità di dati.)
- Dispone di oltre 50 coppie attributo/valore con la possibilità di implementazioni specifiche a cura dei vendors.

- Supporta il modello AAA
- Fornisce supporto per i prodotti NAS (Network Access Service) disponibili in commercio (questo gli assicura un futuro almeno per i prossimi 10 anni).

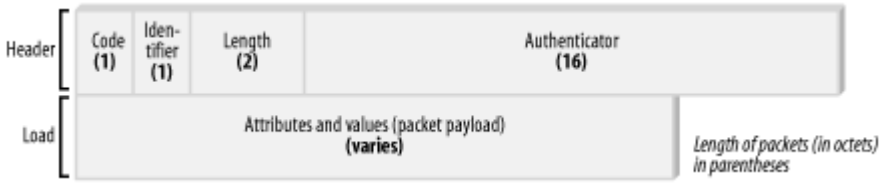
La sicurezza è un ostacolo in alcune implementazioni: l'implementazione di questo protocollo più utilizzata è per i proxy servers, il problema è che i dati di autenticazione devono fare diversi 'salti' sulla rete e questo li rende 'disponibili', se i passaggi non sono sicuri.
Radius non dispone di un supporto per richiamata e deallocazione delle risorse dopo che è stata data l'autorizzazione. E' quindi possibile avere un multi-hop proxy server Radius dove il primo garantisce l'accesso e contatta l'equipment per fornire il servizio, se per qualunque ragione questo servizio non è disponibile, non è indicato nell'RFC alcuna specifica di negazione e disconnessione dal servizio.
Il fatto che sia state-less è limitativo, in quando non tiene traccia dei settaggi di configurazione o informazioni per sessioni successive.
Si evidenzia un problema di scalabilità, cioè un degrado di prestazioni se viene usato su larga scala: non gestisce i problemi di congestione.

2.2 - SPECIFICHE RADIUS

Radius utilizza UDP; chiede che le interrogazioni al server di autenticazione primario, nel caso non vadano a buon fine, siano girate su un server secondario, così una copia di questo deve esistere al livello trasporto del modello OSI; questo consente l'uso di un timer per la ritrasmissione. E' multithread, permette di accettare molte richieste contemporaneamente.
Il server radius risponde alla porta 1812, anche se in un precedente RFC la porta era la 1645.

2.3 - PACCHETTO

Struttura pacchetto Radius:



La struttura è divisa in cinque aree principali:

Code > 1 ottetto, definisce il tipo di messaggio che trasporta il pacchetto. Può contere i seguenti valori:

'1' – Access-Request
'2' – Access-Accept
'3' – Access-Reject
'4' – Accounting-Request
'5' – Accounting-Response
'11' – Access-Challenge
'12' – Status-Server

‘13’ – Status-Client
‘255’ – riservato

Identifier > un ottetto, serve per gestire threading o per fornire un legame diretto tra la richiesta iniziale e le successive repliche. Il server Radius intercetta generalmente i messaggi duplicati esaminando alcuni fattori come l’indirizzo IP, la porta UDP sorgente, il tempo che intercorre tra messaggi sospetti, il valore del campo Identifier.

Lenght > due ottetti, specifica la lunghezza del messaggio Radius (tra 20 e 4096 ottetti). Il campo è testato quando il server riceve il pacchetto al fine di assicurare l’integrità dei dati.

Authenticator > solitamente di 16 ottetti. Si utilizza per verificare e ispezionare l’integrità del payload. Ci sono due specifici tipi di valori del campo Authenticator: i valori di richiesta e i valori di risposta. Il valore di richiesta Authenticator è usato in pacchetti di Authenticator-Request e Accounting-Request. Nella request il campo è un codice random. Il valore di risposta è usato per pacchetti Access-Accept, Access-Reject e Access-Challenge. Si ottiene usando MD5 che genera un hash dai valori di code + identifier +lunghezza + request-autenticator + attributi + shared secret.

2.4 - TIPI DI PACCHETTO

Ci sono quattro tipi di pacchetto che assumono rilevanza nelle fasi di autenticazione e autorizzazione:

- Access-Request:
  E’ usato dal servizio client nel momento in cui spedisce un pacchetto di richiesta di un particolare servizio della rete.

Pacchetto di richiesta di accesso:

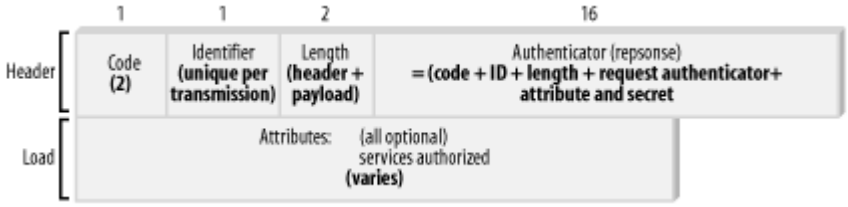
Code	1
Identifier	Unique per request
Length	Header length plus all additional attribute data
Authenticator	Request
Attribute Data	3 or more 4

- Access-Accept

  Spedito dal server Radius per dire al client che la sua richiesta è stata accettata. Il client, dopo aver ricevuto il pacchetto di Access-Accept, a questo punto fa il match tra il campo Identifier ricevuto in questo pacchetto e quello da lui inviato nel pacchetto di Access-Request.

Pacchetto di richiesta accettata

Code	2
Identifier	Identical to Access-Request per transaction
Length	Header length plus all additional attribute data
Authenticator	Response
Attribute Data	0 or more

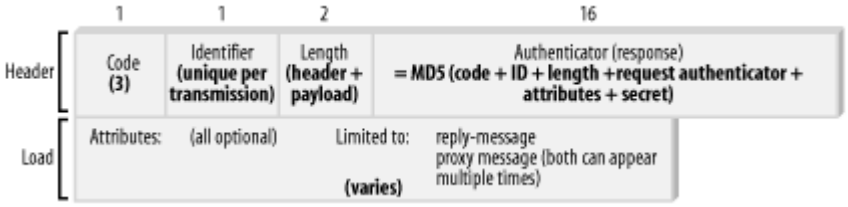


- Access-Reject

  Usato dal server nel caso voglia spedire al client la negazione dei servizi richiesti a causa delle policies di sistema, privilegi insufficienti o altri criteri. Può tuttavia essere inviato in qualunque momento per chiudere una connessione che ha raggiunto il suo limite di tempo. Il payload del pacchetto è limitato a due specifici attributi: Reply- Message e Proxy-State.

Pacchetto di connessione rifiutata

Code	3
Identifier	Identical to Access-Request
Length	Header length plus all additional attribute data
Authenticator	Response
Attribute Data	0 or more

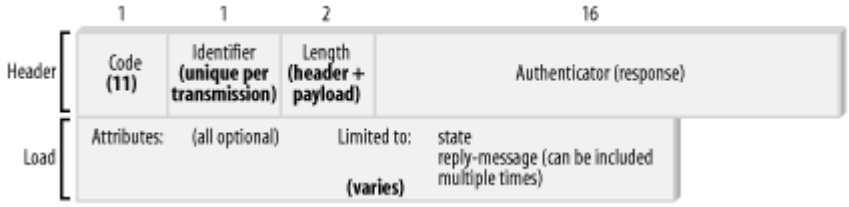


- Access-Challenge

  Se il server riceve informazioni conflittuali dall'utente, per garantire sicurezza, può inviare questo tipo di pacchetto che richiede ulteriori informazioni al client. Quest'ultimo a sua volta invierà un successivo Access-Request con le appropriate informazioni richieste.

Pacchetto di richiesta credenziali

Code	11
Identifier	Identical to Access-Request
Length	Header length plus all additional attribute data
Authenticator	Response
Attribute Data	0 or more



2.5 - SHARED SECRET

Per rendere più forte la sicurezza e migliorare l'integrità delle transazioni, il protocollo Radius utilizza il concetto di shared secret. Lo shared secret è un valore random condiviso tra client e server. L'RFC raccomanda che non sia più piccolo di 16 ottetti. Se un utente finale sottoscrive la sua connessione con molti ISP, fa indirettamente richiesta a molti server Radius: i segreti condivisi tra il client NAS equipment dei diversi ISP, che sono usati per comunicare con i rispettivi server Radius, sono distinti.

Esiste tuttavia un problema nell'utilizzo e conseguente necessaria modifica, di un segreto condiviso: non c'è garanzia che, client e server, possano sincronizzarsi contemporaneamente sul nuovo segreto condiviso nei tempi opportuni (ad es. perché il client è impegnato in altri processi).

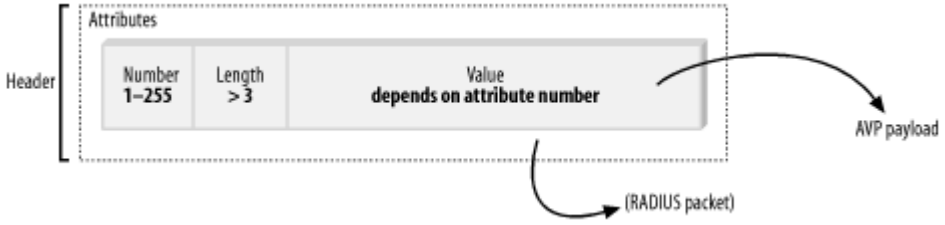
2.6 - ATTRIBUTI E VALORI

L'intera transazione Radius è costruita attorno alle AVPs (Attribute-value pairs), sono coppie di tuple <attributo-valore>. I nomi degli attributi non sono passati nel pacchetto, ma viene passato sono un numero che li identifica.

Gli attributi descrivono i comportamenti e le proprietà dei servizi; sono trasmessi all'interno del pacchetto Radius in un formato standard predeterminato e sono caratterizzati da 3 elementi:

- Numero (denota il tipo di attributo, il range va da 1 a 255)
- Lunghezza del campo attributo
- Valore (proprietà)

Standard AVP transmission pattern

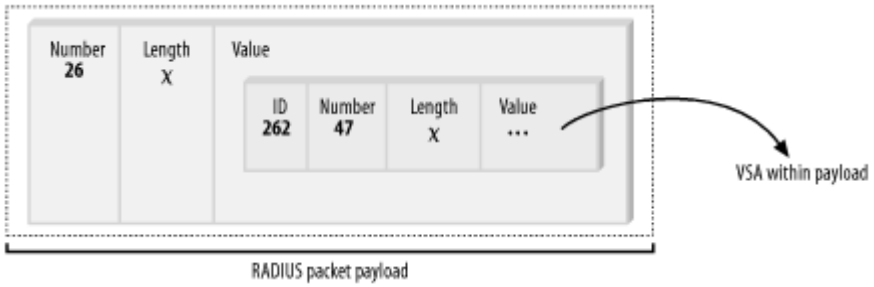


La struttura AVP consiste in una sequenza di byte contenenti ognuna almeno 3 ottetti. Il server Radius conosce la corrispondenza tra il numero di attributo e il suo nome. L'RFC contiene la guida ufficiale dei nomi attributo, i vendors tuttavia possono modificare i nomi nelle loro implementazioni.
Gli attributi possono essere di tipo: integer (INT), Enumerated (ENUM), ip address (IPADDR), character string con codifica utf-8 (STRING), un valore che indica il numero di secondi decorsi dall' 1/1/1970 (DATE) e BYNARY.

2.7 - ATTRIBUTI VSA

Il protocollo Radius è molto flessibile in quanto i vendors possono definire attributi specifici per le loro implementazioni. Alcuni vendors come US Robotics/3Com non seguono addirittura le specifiche RFC.
Il protocollo definisce un particolare AVPs come gateway, dove gli attributi degli specifici venditori (VSAs) possono essere incapsulati. I VSAs sono incapsulati nel payload dello standard AVP (con Numero 26) Vendors-specific.

Passaggio VSA attraverso standard AVP



Vendor ID > 4 ottetti che rappresentano il VSA (sviluppatore/disegnatore/proprietario). Qui '262' ("Formation, Inc.") è l'Assignment Number definito nella tabella PRIVATE ENTERPRISE NUMBERS - SMI Network Management Private Enterprise Codes, dell' RFC 1700.
Vendor type > valore compreso tra 1 e 255, il cui significato dipende dalle specifiche del server in cui è implementato il protocollo Radius di quello specifico vendors.
Length > lunghezza intero VSA
Value > valore della specifica implementazione

2.8 - VALORI DEGLI ATTRIBUTI

Ogni attributo ha un valore, anche se questo è null. I valori rappresentano le informazioni che ogni particolare attributo trasporta.

Esempi di tipi di attributo e valore atteso nel payload di ognuno:

Attribute type	Length (in octets)	Size/Range	Example payloads
Integer (INT)	4	32-bit unsigned	6 256 2432 65536
Enumerated (ENUM)	4	32-bit unsigned	3 = Callback-Login 4 = Callback-Framed 13 = Framed-Compression 26 = Vendor-Specific
String (STRING)	1-253	Variable	"Charlotte" "Raleigh" "206.229.254.2" "aslyterdesign.com"
IP Address (IPADDR)	4	32-bit	0xFFFFFE 0xC0A80102 0x1954FF8E 0x00000A
Date (DATE)	4	32-bit unsigned	0xC0A80102 0xFFFFFE 0x00000A 0x1954FF8E
Binary (BINARY)	1	1 bit	1

2.9 - DIZIONARIO

La macchina Radius server deve disporre di un dizionario che gli permette di collegare il numero di attributo che viaggia nel pacchetto con l'attributo corrispondente. Il dizionario può essere in un file testo, in un database, l'importante è che le informazioni siano accessibili dal server Radius.

Esempio di dizionario:

```
#   ATTRIBUTE-NAME          TYPE
#--------------------------------------
1  User-Name                STRING
2  User-Password            STRING
3  CHAP-Password            STRING
4  NAS-IP-Address           IPADDR
5  NAS-Port                 INT
6  Service-Type             ENUM
7  Framed-Protocol          ENUM
8  Framed-IP-Address        IPADDR
9  Framed-IP-Netmask        IPADDR
10 Framed-Routing           ENUM
.....
```

Considerando l'attributo n. 7 (Framed-Protocol) vediamo come il server Radius che riceve come attributo '7' conosce anche la relativa tabella di corrispondenza del valore contenuto:

```
# VALUE-MEANING                    FOR ATTRIBUTE
#-------------------------------------------
1 PPP                                 7
2 SLIP                                7
3 AppleTalk Rem. Acc. Protocol (ARAP) 7
4 Gandalf SingleLink/MultiLink        7
5 Xylogics proprietary IPX/SLIP       7
6 X.75 Synchronous                    7
```

2.10 - METODI DI AUTENTICAZIONE

Radius supporta una varietà di diversi meccanismi di protocollo per trasmettere i dati specifici tra gli utenti e l'authentication server. I più comuni sono PAP (Password Autentication Protocol) e CHAP (Challenge Handshake Authentication Protocol). Tuttavia Radius permette l'utilizzo di ulteriori attributi e metodi sviluppati dai vendors, inclusi supporti per peculiarità di WinNT, Win2000 e altri popolari sistemi operativi e directory services.

2.10.1 – PAP

L'attributo password utente in un pacchetto di richiesta segnala al server Radius che sarà utilizzato il protocollo PAP per quella transazione. Qui l'unico campo obbligatorio è quello della password utente. Il campo nome utente non deve necessariamente essere incluso in un pacchetto di richiesta. E' possibile che il Radius server, se fa attività di proxy, possa cambiare il valore contenuto nel campo nome utente. Per nascondere l'originale password l'algoritmo è complesso: il client applica MD5 all'identificatore e al segreto condiviso. La password originale viene passata attraverso un processo di xor e il risultato di queste due combinazioni è messo nel campo user-password. Il server Radius ricevente fa il reverse per determinare se il client è autorizzato alla connessione.

2.10.2 - CHAP

E' basato sul concetto che nessuna password deve essere spedita nei pacchetti in rete. Chap cifra in maniera dinamica l'ID del richiedente e la sua password. L'utente riceve una chiave dal suo Radius client. Applica un algoritmo di hash alla chiave ricevuta e rispedisce il chap ID, una chap response e il nome utente del suo radius client. Il Radius client mette il chap id nell'attributo chap-password e risponde. Il challenge value originale ottenuto è nell'attributo chap-challenge o nell'authentication field dell'header, così il server può facilmente accede a questo valore per autenticare l'utente.

ll server Radius usa il valore Chap-Challenge: il chap ID e la password assegnata all'utente e cifrata con un algoritmo di hash (MD5) Il risultato di questo algoritmo deve essere identico al valore che riceve nell'attributo Chap-Password.

Radius non è limitato dall'autenticazione PAP o CHAP. Il limite del protocollo di autenticazione è nel S.O. Ad esempio Radius può supportare l'attributo del dominio quando ci si logga su window NT o Win2000.
Il fattore chiave nel supportare l'autenticazione Radius è come la password debba essere disponibile al sistema host. Il modo più comune per fare questo è usare un file password Unix, ma questo particolare file può lavorare solo con autenticazione PAP. Le password possono essere richiamate da directory service (come active directory di Microsoft, oppure una generica directory di memorizzazione LDAP o usando altri metodi). Comunque il supporto per i vari authenticator protocols dipende dalla particolare configurazione che viene fatta per Radius.

2.11 - REALMS IDENTIFIER

Radius è flessibile, in quanto può supportare diversi modelli organizzativi e infrastrutturali.
Consideriamo l'esempio di tre Internet Service Provider regionali: A, B, C. Si considera che la maggior parte delle risorse a cui i clienti accedono sono confinate nella particolare area di competenza di ognuno. I 3 ISP decidono di concedersi tra loro accessi nelle rispettive aree di competenza: ognuno offre servizio di roaming per gli altri due. Ogni ISP ha necessità che gli utenti autorizzati possano connettersi e contemporaneamente devono proteggere i dati sensibili dei propri clienti. Radius, supporta, per identificare gli utenti: "REALMS", si tratta di una stringa posta prima o dopo il valore che, nel campo che è assegnato allo user-name e serve per indicare l'identità del server da contattare per iniziare il processo AAA. Ad esempio un tipo comune di Realm identifier è conosciuto come prefix-realm: dove il nome dell'ISP è indicato prima del nome utente e separato a questo da un carattere che può essere '@', '\'o, oppure '/'.
Ad esempio al sig.Rossi collegato all'ISP pippo, verrà configurata la propria connessione in modo che passi : "pippo\rossi".

2.12 - ATTRIBUTI STANDARD

Gli attributi standard per Radius secondo gli RFC: sono 63 e forniscono supporto e configurazione per ogni tipo di connessione, terminale virtuale, limiti di tempo per connect/session, filtraggio pacchetti e call-return services. Gli attributi per processi di autenticazione e autorizzazione di una transazione Radius vanno dal numero 1 al 39 e dal 60 al 63. Gli attributi invece relativi all'accounting vanno dal 40 al 59.

Attributi per processi autenticazione e autorizzazione di una transazione Radius (RFC 2138):

1	User-Name
2	User-Password
3	CHAP-Password
4	NAS-IP-Address
5	NAS-Port
6	Service-Type

7 Framed-Protocol
8 Framed-IP-Address
9 Framed-IP-Netmask
10 Framed-Routing
11 Filter-Id
12 Framed-MTU
13 Framed-Compression
14 Login-IP-Host
15 Login-Service
16 Login-TCP-Port
17 (unassigned)
18 Reply-Message
19 Callback-Number
20 Callback-Id
21 (unassigned)
22 Framed-Route
23 Framed-IPX-Network
24 State
25 Class
26 Vendor-Specific
27 Session-Timeout
28 Idle-Timeout
29 Termination-Action
30 Called-Station-Id
31 Calling-Station-Id
32 NAS-Identifier
33 Proxy-State
34 Login-LAT-Service
35 Login-LAT-Node
36 Login-LAT-Group
37 Framed-AppleTalk-Link
38 Framed-AppleTalk-Network
39 Framed-AppleTalk-Zone
60 CHAP-Challenge
61 NAS-Port-Type
62 Port-Limit
63 Login-LAT-Port

Gli attributi 34, 35 e 36 sono specifici per il supporto Radius nelle LAT (local area transport). Permettono al client di creare una connessione terminale virtuale usando una connessione asincrona sopra qualunque tipo di Lan.

2.13 - ACCOUNTING

Spesso gli ISP devono gestire la loro presenza in diverse locazioni. In ognuna di queste sedi devono garantire un'adeguata protezione e una linea di difesa robusta per autenticazione (verifica dell'identità dichiara dal client), e autorizzazione, al fine di fornire all'utente solo i servizi ai quali può accedere. Importante è comunque sapere quali sono gli utenti che si loggano, cosa fanno e quali servizi usano: tutte informazioni che si ottengono dal processo di accounting.
L'accounting in Radius è basato su tre fondamentali elementi:

- Modello client-server: la macchina su cui avviene l'accounting è il server Radius al quale il client Radius si connette. Il client passa i dati al server per essere processati. Il server segnala il corretto ricevimento dei dati. E' anche possibile per il server Radius agire come accounting proxy.

- La comunicazione tra i vari device deve essere sicura: tutte le informazioni tra server e client vengono passate attraverso l'uso dello shared secret.
- Radius accounting è estensibile: il formato degli attributi di accounting è simile a quello degli attributi di autenticazione e autorizzazione. La maggior parte dei servizi offerti dalle varie implementazioni vengono definiti usando le AVPs.

2.14 - OPERAZIONI DI ACCOUNTING

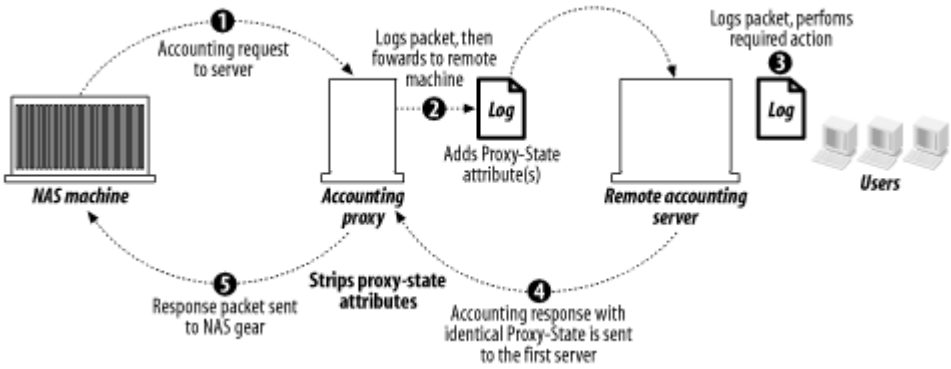
Ogni comunicazione relativamente al Radius accounting avviene attraverso un pacchetto di Accounting-Request: il cliente genera un pacchetto di Accounting-Start con le informazione circa i servizi che richiede e la propria identità. Il server fa l'acknowledge del ricevimento della richiesta. Quando il client ha finito invia un pacchetto di Accounting Stop, che contiene informazioni circa il servizio utilizzato, il tempo, la quantità di dati trattati, la velocità,.. Il server restituisce conferma delle informazioni ricevute. Se il server non risponde ai pacchetti di Accounting-Request del client, l'RFC consiglia all'utente di ritentare finché non si riceve l'acknowledge dal server. Infatti in una grossa rete distribuita, è desiderabile avere più di un accounting server in modo da garantire una certa ridondanza.

Il processo di RADIUS accounting proxies:

1. Il RADIUS client spedisce un pacchetto Accounting Start all'accounting server.
2. L'accounting server ricevente fa il log del pacchetto. Può aggiungere o modificare attributi e invia il pacchetto ad una macchina remota.

La macchina remota fa il log e spedisce il pacchetto di Accounting-Response indietro al server che gli ha inviato la richiesta. Il server di partenza riceve l'acknowledgment, toglie le informazioni relative alle attività di proxy, costruisce e aggiunge il Response Authenticator, e spedisce il pacchetto al client.

Processo di proxying per accounting RADIUS



2.15 - IL PACCHETTO DI ACCOUNTING

Secondo l'RFC 2866 il pacchetto Radius accounting è incapsultato in un pacchetto UDP e la porta di destinazione è la 1813.

Code > contiene '4' se Accounting-Request e '5' se Accounting-Response, se il valore è diverso viene scartato senza alcuna risposta.
Identifier > un ottetto, per attività di threading
Length > 2 ottetti, lunghezza pacchetto Radius per garantire integrità (il pacchetto deve avere lunghezza tra i 20 e 4095 ottetti: se più lungo la parte eccedente viene ignorata, se più corto viene subito scartato.
Authenticator > generalmente 16 ottetti. Qui l'integrità del pacchetto payload è ispezionata e verificata.
Ci sono due tipi di Authenticator:

- Request authenticator: 16 ottetti, contiene il check sum MD5, viene calcolato su code, identifier, length, attributes, shared secret.
- Response authenticator: si tratta sempre di un hash (usando MD5), calcolato usando code, identifier, length, request authenticator dalla richiesta originale e gli attributi della response.

Il pacchetto di accounting non può essere sicuro al 100%, perché ad esempio il cliente che non riceve l'acknowledge o una risposta dal server, continua a spedire la propria request per un tempo potenzialmente illimitato, questo rende possibile il fatto che in alcune sessioni possono esserci record inconsistenti. Ci sono due tipi di pacchetti Radius che sono rilevanti per la fase di accounting in una transazione AAA.

- Pacchetto Accounting Request:

Code	4
Identifier	Unique for each request; unique for each transmission of modified data
Authenticator	Request
Attribute Data	0 or more attributes

- Pacchetto Accounting Response:

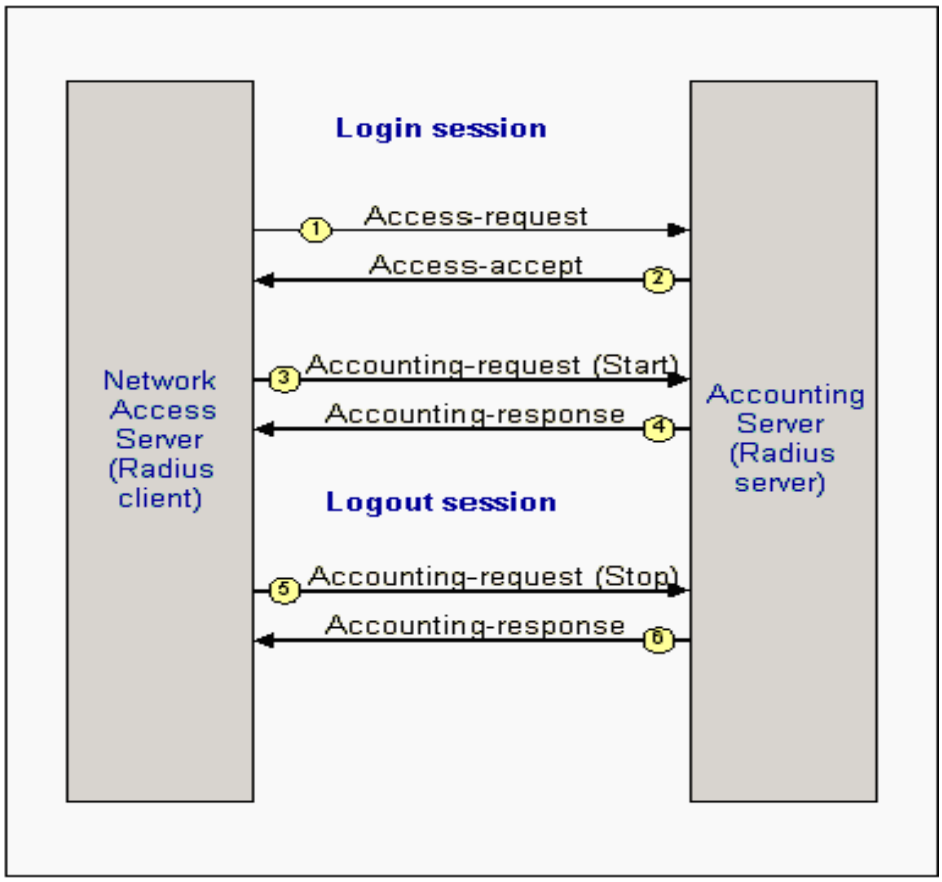
Code	5
Identifier	Identical to corresponding Accounting-Request
Authenticator	Response
Attribute Data	0 or more attributes

E' il pacchetto di acknowledge inviato dal server al client e indica che la richiesta è stata ricevuta e loggata. L'Identifier è lo stesso dell'accounting request.

Gli attributi di accounting:

40 Acct-Status-Type
41 Acct-Delay-Time
42 Acct-Input-Octets
43 Acct-Output-Octets
44 Acct-Session-Id
45 Acct-Authentic
46 Acct-Session-Time
47 Acct-Input-Packets
48 Acct-Output-Packets
49 Acct-Terminate-Cause
50 Acct-Multi-Session-Id
51 Acct-Link-Count

1.16 - Un esempio di scambio di messaggi RADIUS:



3 - APPLICATIVI

3.1 - FREE RADIUS

E' distribuito sotto licenza GNU General Public License (GPL - Libertà di download e installazione). Fornisce supporto per:

- limitare il numero massimo di connessioni contemporanee
- permettere e negare accesso agli utenti singolarmente o anche in base alla loro distinzione in gruppi
- rendere possibile una ricerca intelligente di files che selezionano i protocolli di autenticazione sulla base della sintassi del nome utente
- esecuzione di programmi esterni su log-in riuscite
- possibilità di inclusioni di files con configurazioni, utenti e dizionari
- utilizzo attributi specifici dei fornitori
- gestire proxy server Radius
- tutta una serie dei più popolari NAS equipment (3Com, Cisco)

E' disponibile su un'ampia rete di piattaforme (Linux, Free e Open BDS), Unix, Solaris e altri). Usando un database Mysql, l'amministratore può fare interrogazioni e report sulle transazioni usando il linguaggio sql (supportato su qualunque piattaforma), questo consente a utenti e password per autenticazioni di trovare una loro collocazione centralizzata. Per introdurre un livello di fault-tolerance e aggiungere sicurezza all'integrità dei dati è bene considerare l'ipotesi di avere due macchine con Mysql che si replicano il database di autenticazione degli utenti. Le macchine devono avere la stessa versione di Mysql, occorre decidere quale delle due macchine è il master server e quale lo slave (per determinare la direzione della duplicazione). Radius è un protocollo state-less e oltretutto normalmente il server Radius ha una lista interna di chi è connesso e lo stato attuale delle porte client. Molti NAS equipment dispongono di un meccanismo con il quale il demone Radius di authentication request può fare interrogazioni per trovare quale utente è assegnato a quale porta: lo si fa attraverso SNMP (Simple Network Monitoring Protocol).

L'utilizzo di Radius per autenticazione via web, per proteggere l'accesso ad un sito, è implementabile: è quindi possibile configurare Apache (parlando di realtà nel mondo open-source) in modo da permettere autenticazione su un Radius database utenti. Apache diventa un client Radius (occupando la posizione tradizionale del NAS nel canale di autenticazione).

FreeRadius supporta LDAP.

3.2 - Cistron RADIUS server (http://www.radius.cistron.nl/)

- licenza GNU GPL (l'autore è Miquel van Smoorenburg - http://www.miquels.cistron.nl/) .
- su sistemi operativi: Linux, FreeBSD, OpenBSD, OSF/Unix, Solaris.
- Può esse utilizzato un database per gestire gli utenti.
- E' in grado di prevenire il login doppio.
- Mantiene la lista degli utenti loggati.
- Supporta gli specifici Vendor-Spcific attributes.
- Consente attività di proxing.
- Consente la replicazione dei dati di account tra servers.

3.3 - ClearBox Server

- software proprietario (XPerience Technologies – http://www.xperiencetech.com).
- richiede sistema operativo Windows.
- applicatiivo scritto in C++ con ridotto utilizzo di CPU.
- compatibile per il protocollo Radius agli RFC 2865, 2866, 2869, 3579, 3580 e con il protocollo TACACS+.
- fa da proxy server filtrando le richieste e trasmettendole a un server remoto.
- supporta ogni tipo di attributo Vendor-Specific.
- gestisce intestazioni Realms multiple.
- è' implementato come Windows system service. Dispone di interfaccia grafica per attività di setup, configurazione, amministrazione e monitoraggio.
- supporta diversi data source (MS SQL Server, MS Access, e la maggior parte dei DBMS SQL-based attraverso driver ODBC o OLE DB).
- tiene traccia delle sessioni utente attive e consente di limitare il numero di login simultanei degli utenti.
- consente autenticazioni PAP, CHAP, MS-CHAP, MS-CHAPv2, ARAP, EAP-MD5.

3.4 - NavisRadius ™

- proprietario (Lucent Technologies - http://www.lucent.com/)
- piattaforme: Sun Microsystems (Solaris SPARC & x86: dalla release 2.7), Linux (Redhat), Hewlett Packard (HP-UX release 11.0, Compaq/DEC TRU-64 UNIX), Windows, Apple (Macintosh OS X: 10.2 & 10.3)
  richiede la presenza della Java Virtual Machine (JVM) Sun.
- permette di creare sofisticate policy AAA e gestirle in maniera semplice.
- supporta limiti di sessione (utenti, realm, DNIS)
- dispone di interfaccia LDAP (legge dati da directory server LDAP)

- se viene utilizzato per gestire un'implementazione radius già esistente, è in grado di leggere vari formati di dati da altri server.
- Syslog logging – dispone di funzioni di log server e client evidenziando informazioni richieste.
- dispone di interfaccia grafica (GUIs) per amministrazione server e utente.

3.5 - Radiator

- proprietario (OSC - Open System Consultants - http://www.open.com.au/)
- piattaforme: Windows (fino alla vers. 2003), Unix, Linux (Red Hat, Debian, Mandrake, SuSE, Lindows, Slackware, Ubuntu etc on Intel, Sparc, PPC, HP-PA etc), Solaris (Intel and Sparc), FreeBSD, NetBSD, SunOS, AIX, IRIX, SCO Open Server, Digital, HP-UX, Mac OS9, Mac OS X, Novell Open Enterprise Server (NetWare) 6.5, VMS.
- supporta autenticazione tramite le più svariatearchitetture: dai files di testo a DBM files, Unix password files, database SQL, Radius servers remoti (proxying), programmi esterni, Active Directory, LDAP, e altri database proprietari di cui si dispone.
- supporta standard 802.1X.
- previene login doppio, supporta ogni tipo di attributo vendors specific.
- dispone di scripts CGI per configurazione, reporting e gestione di utility per i database
- lavora su NASs, VPDN, ADSL and wireless Access Points.

3.6 - Elenco di altri applicativi, a mio avviso comunque interessanti, dei quali ho indicato solo il sistema operativo che ne permette l'utilizzo:

NTX Access (Internet Transaction Services - http://www.itrans.com/)	WINDOWS
DTC Radius (Digital Technologies Corporation - http://www.dtc.co.jp/)	UNIX, WINDOW
VOP Radius (VIRCOM - http://vircom.com/)	WINDOW
RadiusNT (IEA Software, Inc - http://www.iea-software.com/)	WINDOW
Radtac Manager Server (Media Online Italia s.r.l. - http://www.radtac.com/)	WINDOW
Steel-Belted Radius (Funk software - http://www.funk.com/)	UNIX, WINDOW
Shiva (EICON NETWORK- http://www.eicon.com/)	UNIX, WINDOW
Internet Authentication Service (Microsoft)	WINDOWS
IcRadius (http://sourceforge.net/)	UNIX/LINUX

4 - BIBLIOGRAFIA

Jonathan Hassel, J (2002). RADIUS, O'Reilly

[RFC 2138] Remote Authentication Dial In User Service (RADIUS) - April 1997
[RFC 2139] RADIUS Accounting - April 1997
[RFC 2548] Microsoft Vendor-specific RADIUS Attributes - March 1999
[RFC 2865] Remote Authentication Dial In User Service (RADIUS) - June 2000
[RFC 2866] RADIUS Accounting - June 2000
[RFC 2867] RADIUS Accounting Modifications for Tunnel Protocol Support - June 2000
[RFC 2868] RADIUS Attributes for Tunnel Protocol Support - June 2000
[RFC 2869] RADIUS Extensions - June 2000
[RFC 3579] RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP) - September 2003
[RFC 3162] RADIUS and IPv6 - August 2001
[RFC 3580] IEEE 802.1X Remote Authentication Dial In User Service (RADIUS) Usage Guidelines - September 2003

http://www.securitywireless.info
http://web.mit.edu
http://www.webopedia.com
http://www.freeradius.org/
http://www.radius.cistron.nl/
http://www.xperiencetech.com
http://www.lucent.com/)
http://www.open.com.au/