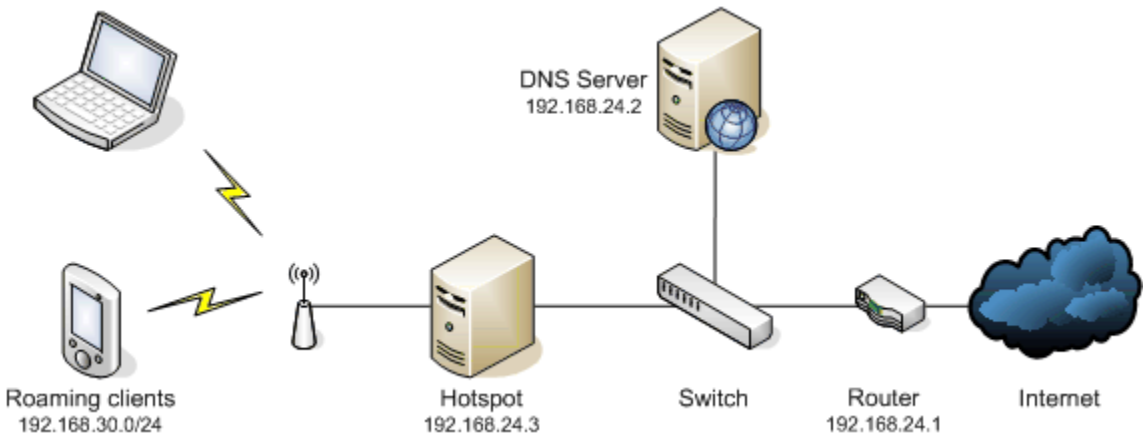
Freeradius

Concetti di base

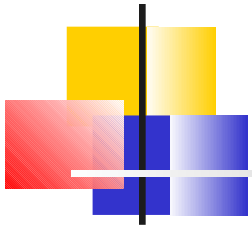
Utilizzo nella vita reale

Esempi di implementazione

Cosa abbiamo visto?



Abbiamo già nominato la parola Radius.



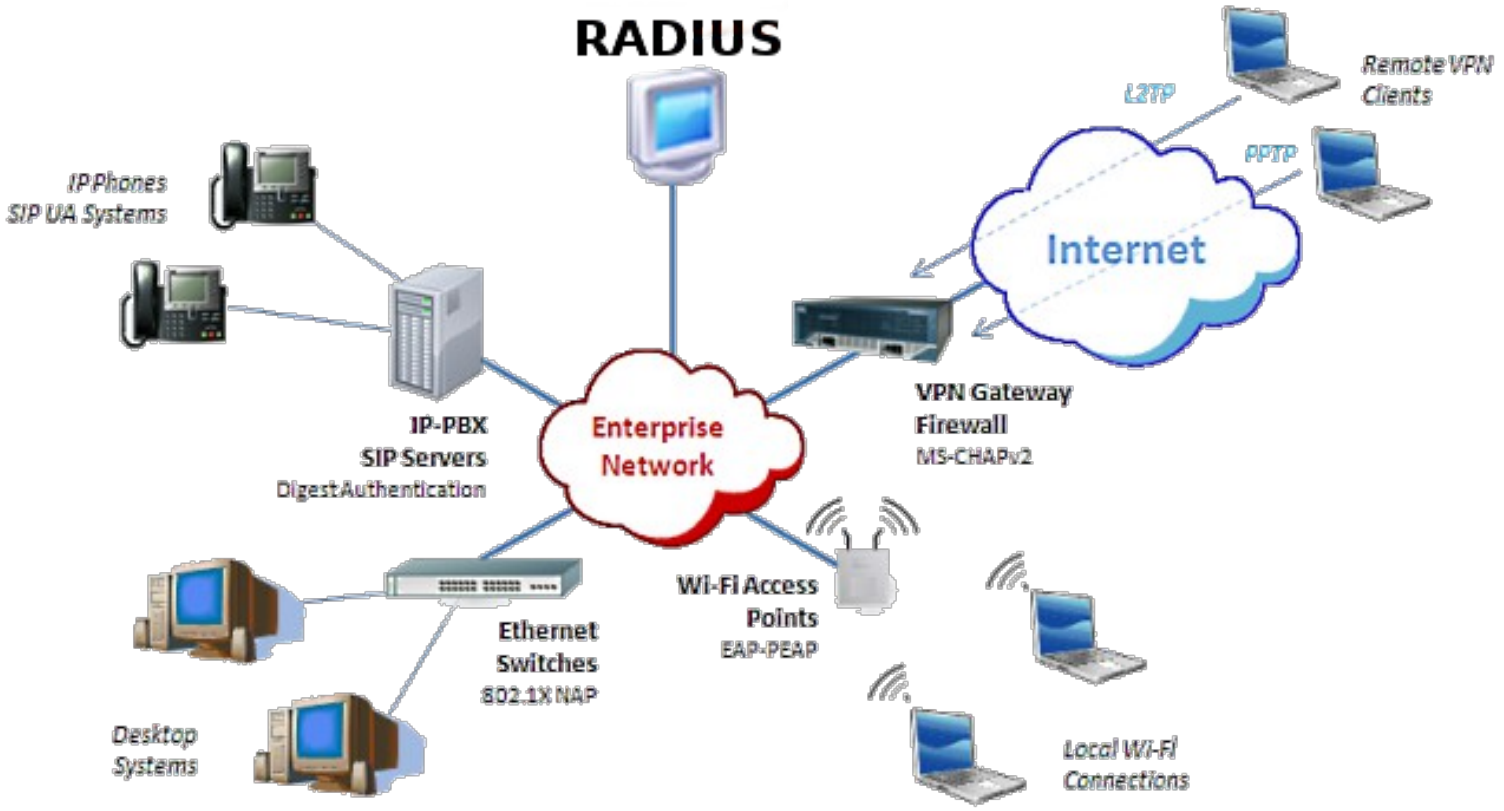
Il protocollo Radius

E' un protocollo di comunicazione AAA.

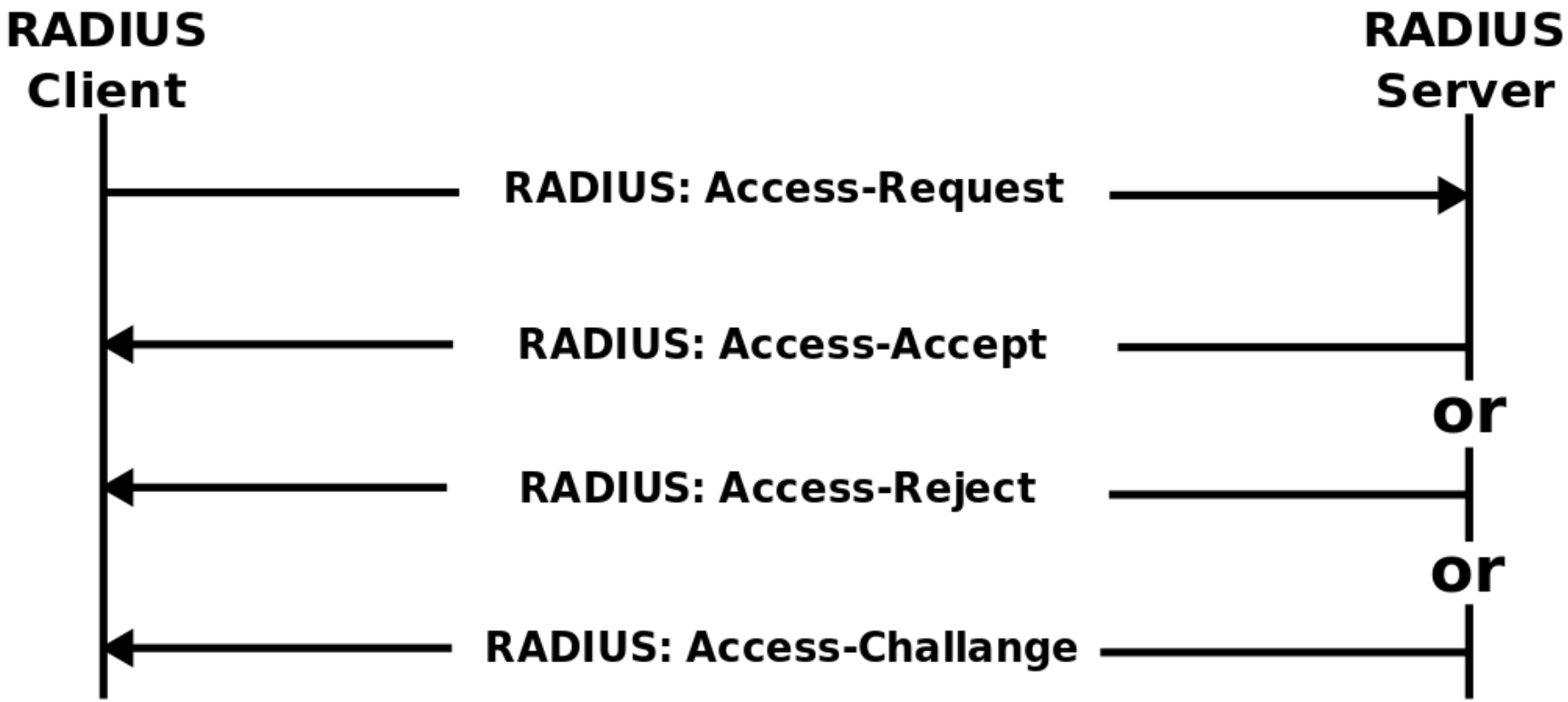
- Authentication
- Authorization
- Accounting

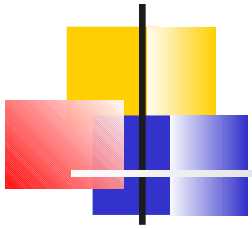
Radius è l'acronimo di:
Remote Authentication **Dial-In** User Service

Dove viene utilizzato?



Autenticazione e Autorizzazine





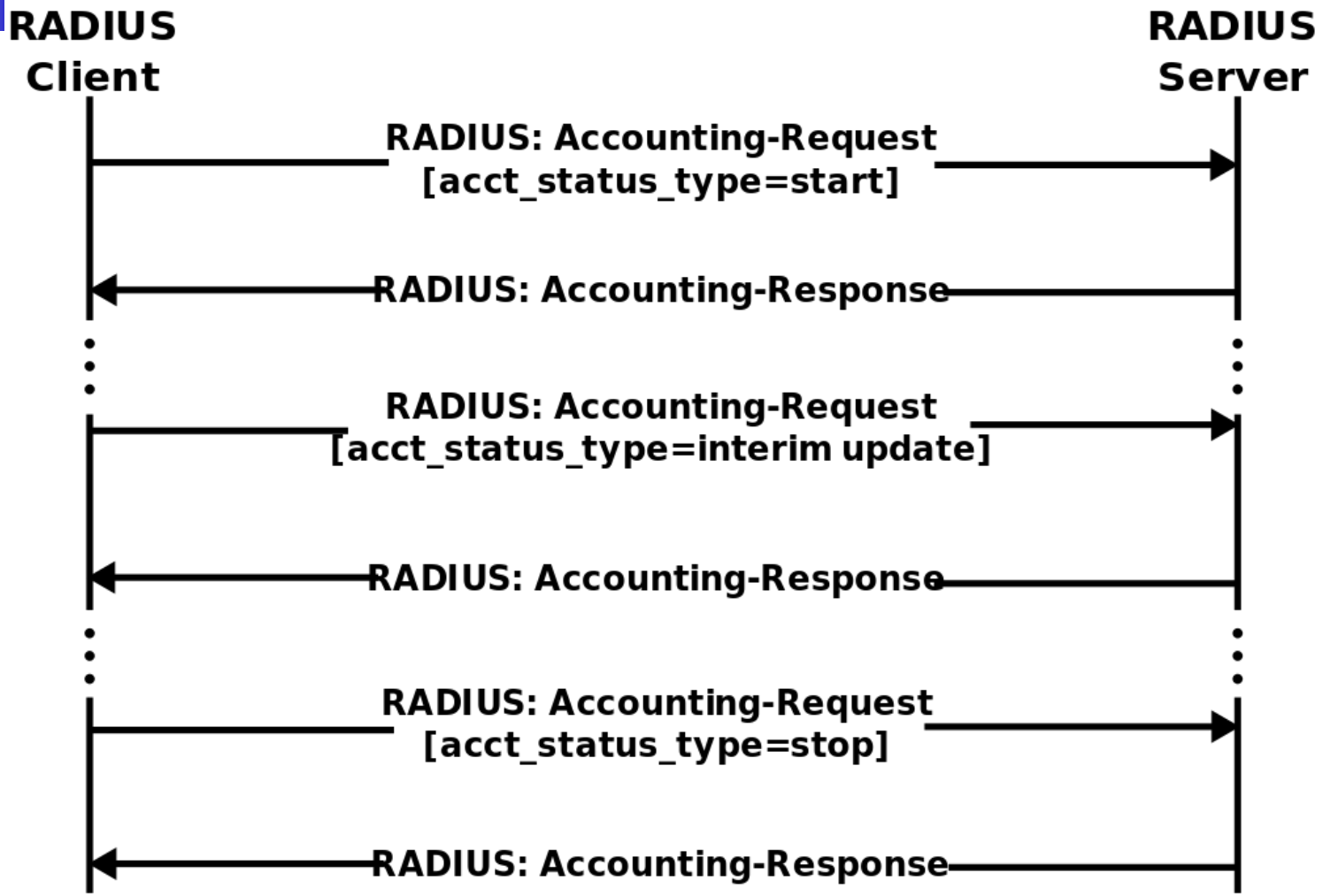
Autenticazione e Autorizzazine

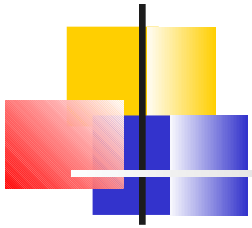
Cosa succede in queste fasi?

- L'utente richiede al NAS/RAS di accedere alla rete
- Il NAS/RAS inoltra la richiesta al server Radius
- In caso di successo il client riceve insieme alla conferma di accesso anche dati utili per la connessione (ad esempio l'indirizzo ip)

NAS: Network Access Server
RAS: Remote Access Server

Accounting


RADIUS Client
RADIUS Server
RADIUS: Accounting-Request
[acct_status_type=start]
RADIUS: Accounting-Response
RADIUS: Accounting-Request
[acct_status_type=interim update]
RADIUS: Accounting-Response
RADIUS: Accounting-Request
[acct_status_type=stop]
RADIUS: Accounting-Response



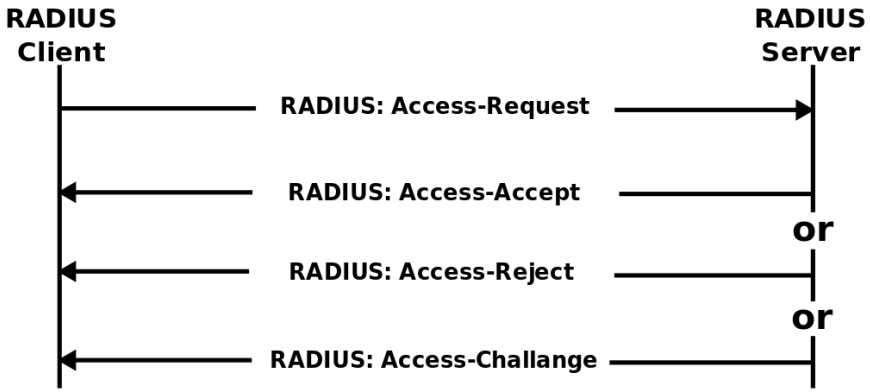
Accounting

Cosa succede in questa fase?

- Il NAS/RAS contatta periodicamente il Radius per aggiornare lo stato della sessione dell'utente.
- Vengono raccolti dati statistici che in futuro possono essere utilizzati per contabilizzare l'utilizzo della rete.
- La sessione viene terminata quando l'utente si disconnette, viene “ucciso” dal server radius o in caso di rilevata inattività.

Chiariamo alcuni dubbi

Chi è il client Radius?



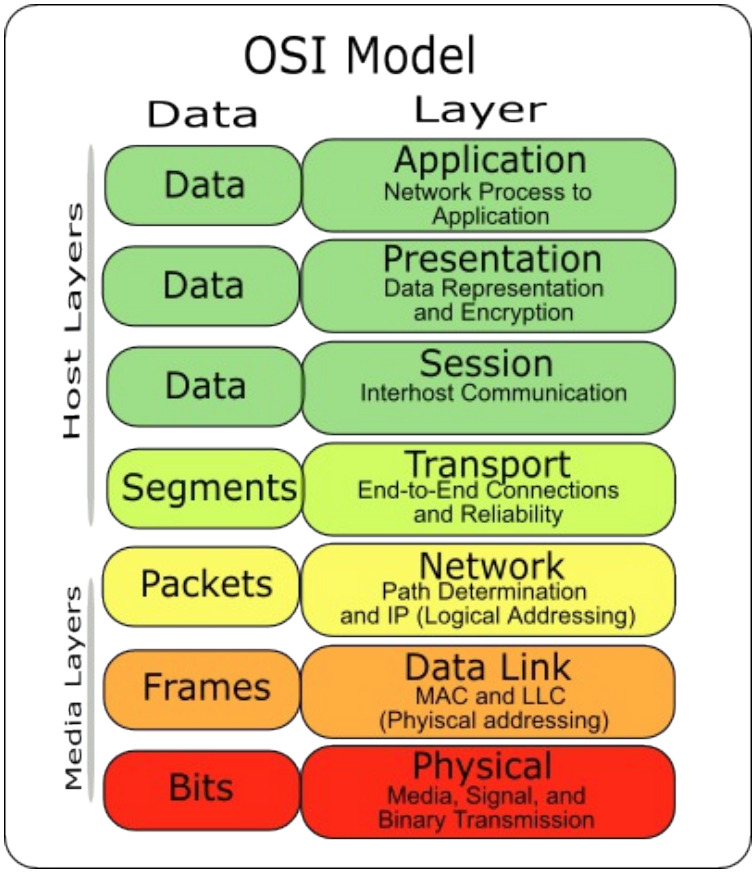
Raramente l'utente parla direttamente con il server Radius.
Questo compito in genere è delegato al NAS/RAS.

Chiariamo alcuni dubbi

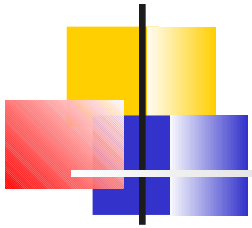
A che livello dello stack ISO/OSI avviene l'autenticazione?

A seconda del protocollo di accesso e dall'implementazione del NAS/RAS, l'autenticazione può venire dal livello di data link fino a quello di applicazione.

Esempi: 802.1X (link), PPP (link), HTTPS (application)



Esempio: access point



Security

This page allow you to transmit your data securely over the wireless network. Matching authentication and encryption methods must be setup on your U.S. Robotics 802.11g Wireless Router and wireless client devices to use security.

WPA (WiFi Protected Access)

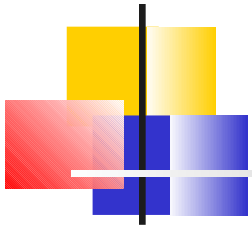
WPA Encryption Type	Enabled - 802.1X Mode

802.1X

Re-Authentication Period	3600	Seconds (0 for no re-authentication)
Quiet Period	60	Seconds after authentication failed

RADIUS Server Parameters:

Server IP	10 . 0 . 0 . 56
Server Port	1812
Secret Key	••••••••••••••••••••••
NAS-ID	Router-Casa



Esempio: access point

Che differenza c'è rispetto ad utilizzare l'autenticazione PSK-TKIP?

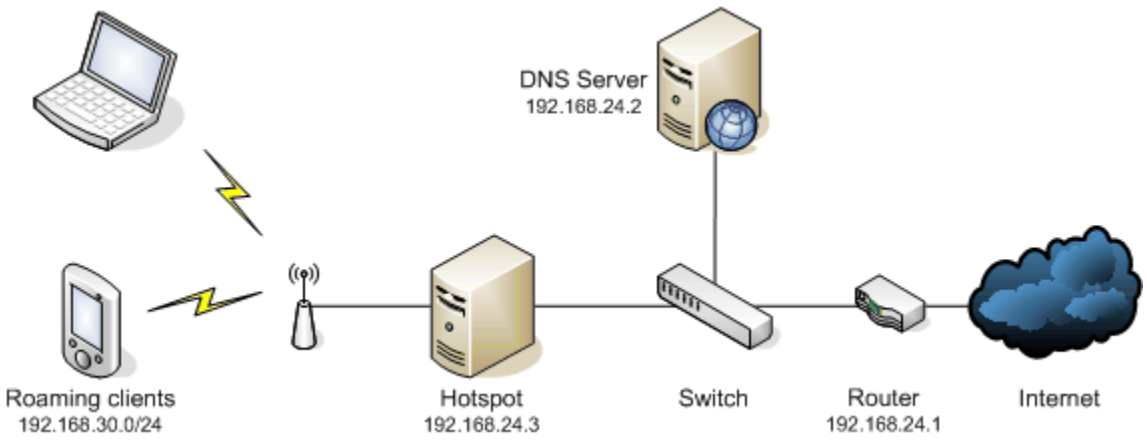
- Ogni utente dispone di una password personale
- Nel caso di violazione o rimozione di una delle password, non è necessario andare a modificare la configurazione degli altri utenti.
- Elimina la necessità di sistemi hotspot, dato che gli utenti devono essere già autenticati per ricevere un indirizzo ip.

Esempio: managed switch

Perché proteggere uno switch?

- Le prese di rete ormai sono ovunque.
- I filtri mac possono essere facilmente aggirati.
- Impedire anche solo la possibilità di accessi non autorizzati in ambienti in cui vi sono dati sensibili è una priorità per la sicurezza.

Esempio: Hotspot

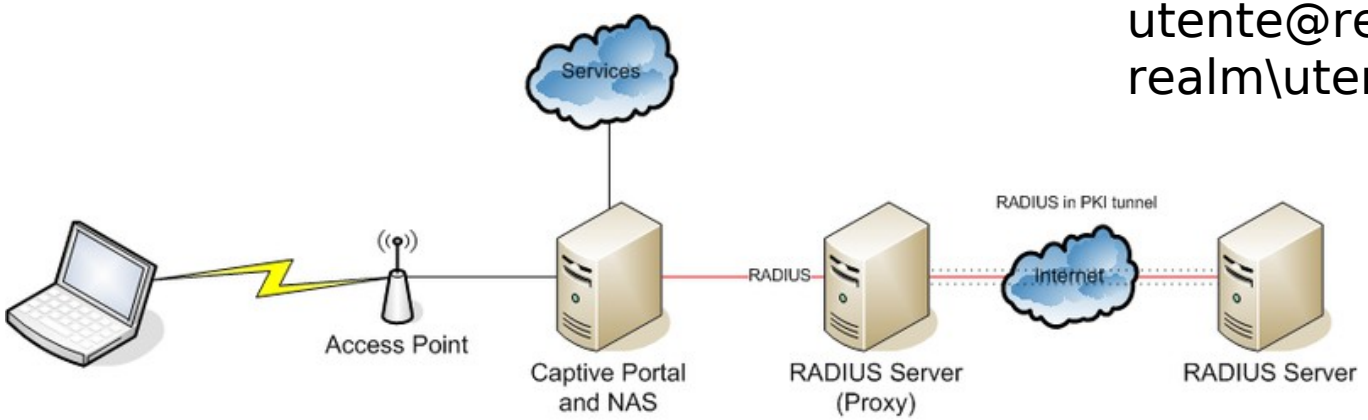


In questo caso, l'utente inizia l'autenticazione utilizzando il protocollo HTTPS.

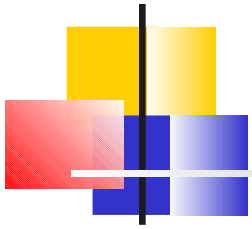
Realms e Roaming

I Realms (Reami) identificano diversi gruppi di utenti, al pari dei domini windows o dei domini delle caselle di posta elettronica.

utente@realm
realm\utente



Roaming: un server radius non autoritativo per un dato reame, può inoltrare la richiesta di autenticazione al server che si occupa dell'autenticazione del reame dell'utente.

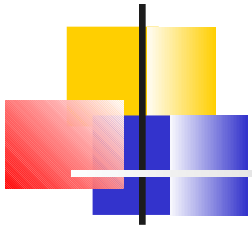


Nessun difetto?

Nessun sistema è perfetto

- Problemi di efficienza legati all'utilizzo del protocollo UDP.
- Vulnerabile agli attacchi di tipo replay.
- Sicurezza garantita per le comunicazioni hop-to-hop e non end-to-end (possibilità di mitm?)

La soluzione? **Diameter!**

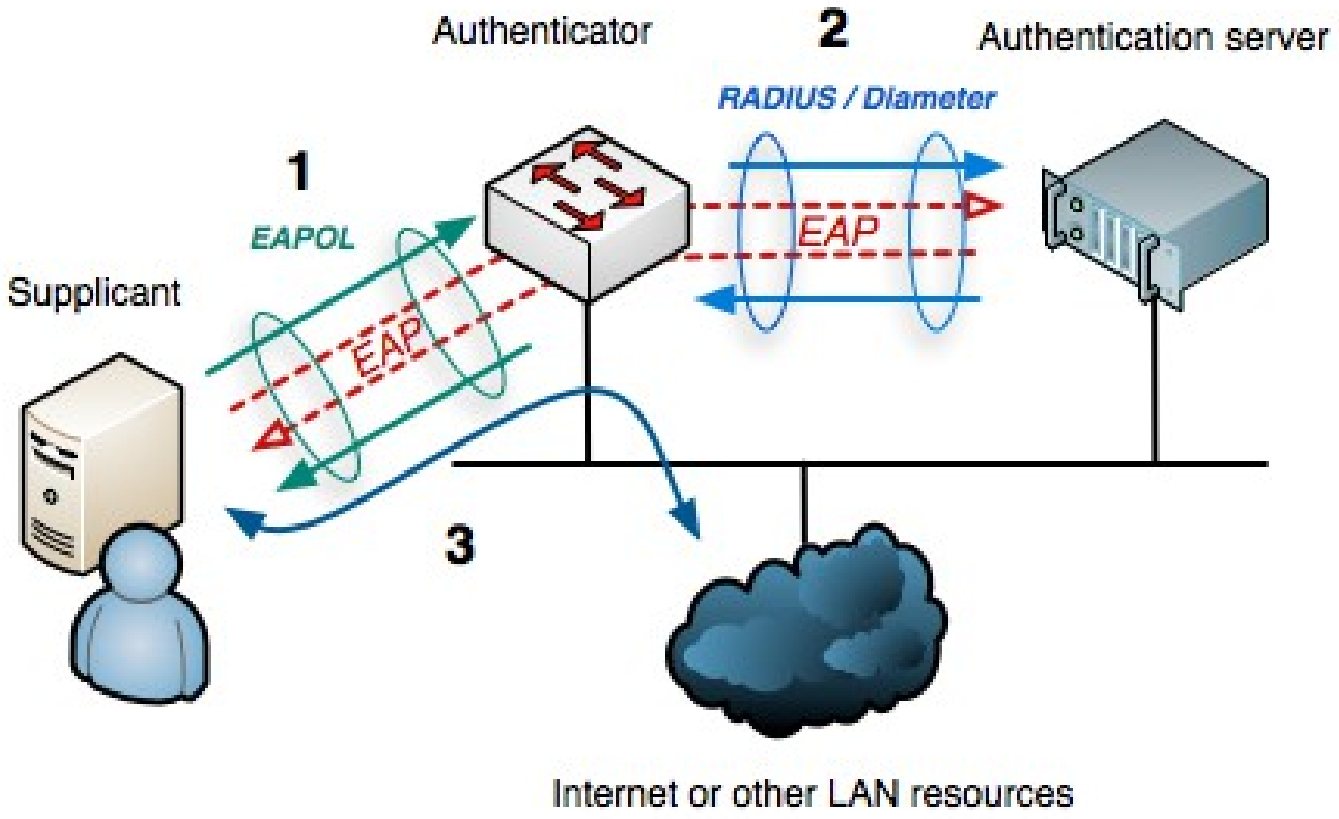


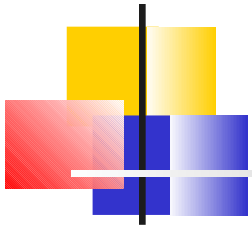
Freeradius

E' la migliore e più famosa implementazione di un server Radius per Linux.

- Supporta tutti i protocolli EAP standard
- Supporta alcuni protocolli proprietari
- Modulare nella configurazione e funzionalità
- Non è un software “for dummies”.

Il protocollo 802.1X

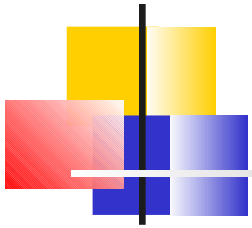




Il protocollo 802.1X

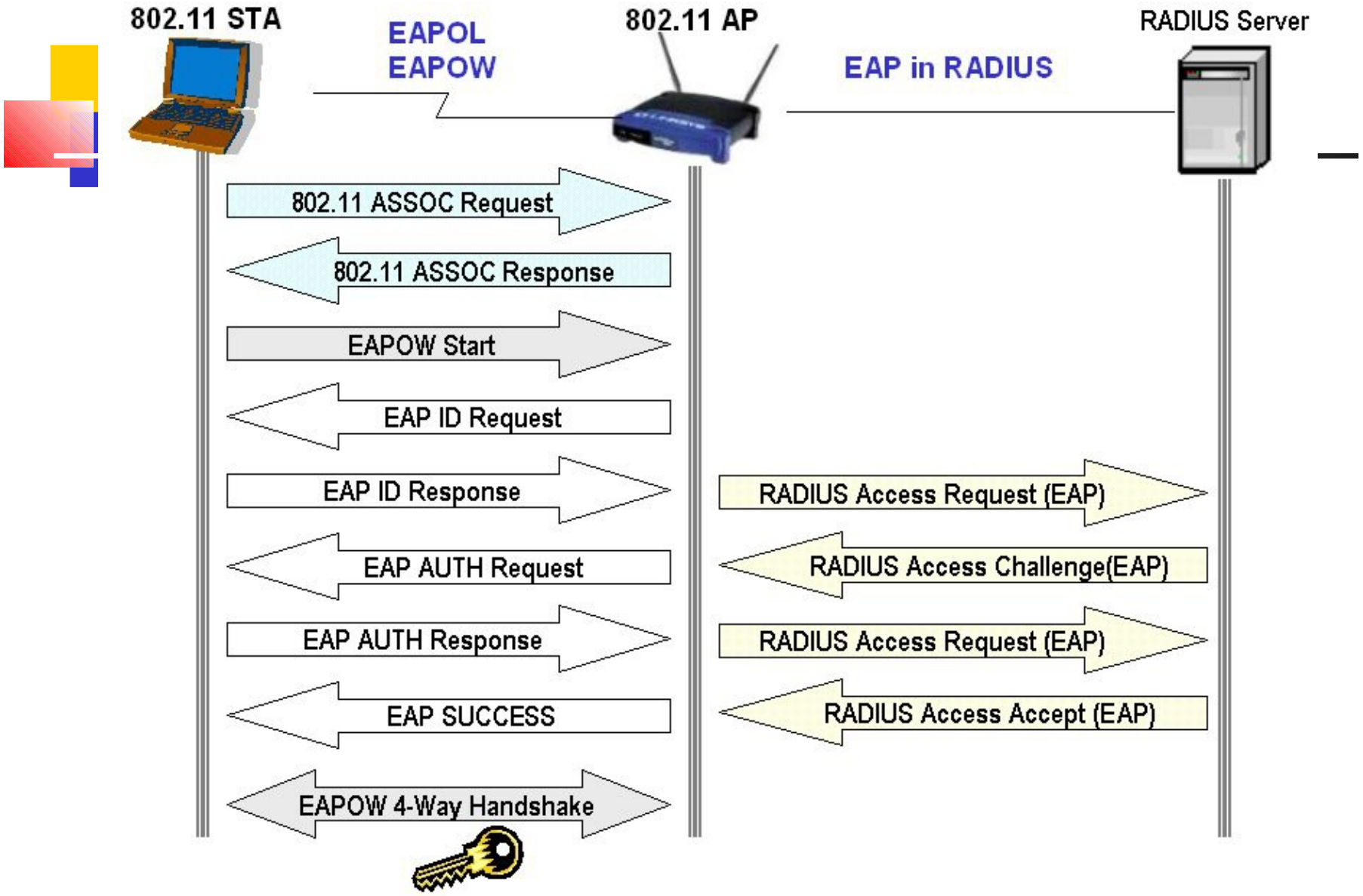
Fasi di autenticazione:

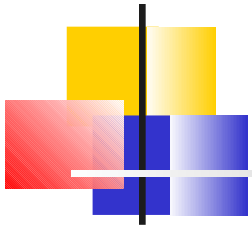
- **Initialization:** l'authenticator marca con lo stato di “unauthorized” una porta o tentativo di accesso wifi quando viene rilevata una connessione. Al supplicant è permesso solo il traffico per il protocollo 802.1x.
- **Initiation:** l'autenticator invia delle richieste *EAP ID Request* alle quali il supplicant risponde con dei frame *EAP ID Response* contenenti informazioni utili quali l'UserID. L'authenticator ricevuto il pacchetto incapsula la risposta in una richesta al radius di tipo *Access-Challenge*.



Il protocollo 802.1X

- **Negotiation:** il radius risponde alla richiesta di tipo Access-Challenge con un pacchetto che contiene la reale modalità EAP da utilizzare per l'autenticazione da rigirare al supplicant. Il supplicant può iniziare la reale autenticazione o “proporre” al sistema una delle modalità che supporta.
- **Authentication:** quando il radius e il supplicant trovano un accordo, supplicant e radius si scambiano i messaggi di Access-Request e relative risposte. Se l'autenticazione ha successo, l'authenticator setta lo stato della port ad “authorized”, altrimenti la lascia nello stato “unauthorized”.


802.11 STA
EAPOL
EAPOW
802.11 AP
EAP in RADIUS
RADIUS Server
802.11 ASSOC Request
802.11 ASSOC Response
EAPOW Start
EAP ID Request
EAP ID Response
RADIUS Access Request (EAP)
EAP AUTH Request
RADIUS Access Challenge(EAP)
EAP AUTH Response
RADIUS Access Request (EAP)
EAP SUCCESS
RADIUS Access Accept (EAP)
EAPOW 4-Way Handshake



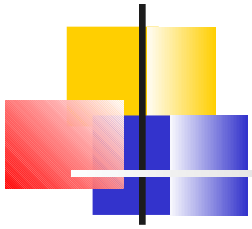
EAP: Extensible Authentication Protocol

EAP è un protocollo di autenticazione.

Il protocollo radius non basta?

- Senza EAP le credenziali verrebbero trasmesse solo utilizzando l'algoritmo di criptazione MD5
- Questa modalità è tutt'altro che sicura!
- Radius, authenticator e supplicant devono parlare la stessa lingua.

Metodo	Chiave dinamica	UserID e password	Metodi di attacco	Commenti
MD5	No	Sì	Attacco basato su dizionario Man in the middle Dirottamento di sessione	Facile da implementare Supportato su molti server Insicuro Richiede database con testo in chiaro
TLS	Sì	No	Offre un'elevata sicurezza	Richiede certificati del client Innalza i costi di manutenzione Autenticazione a due fattori con smart-card
LEAP	Sì	Sì	Attacco basato su dizionario	Soluzione proprietaria Gli access point devono supportarlo
TTLS	Sì	No	Elevata sicurezza	Creazione di un tunnel TLS sicuro Supporta i tradizionali metodi di autenticazione: PAP, CHAP, MS-CHAP, MS-CHAP V2 L'identità dell'utente è protetta (crittata)
PEAP	Sì	Sì	Media sicurezza	Simile all'EAP-TTLS Creazione di un tunnel TLS (SSL) sicuro L'identità dell'utente è protetta (crittata) Attacco su dizionario per le credenziali

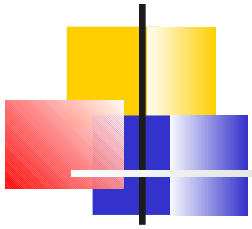


Configurare i client

Per CoovaChilli abbiamo impostato le seguuenti righe nel file /etc/freeradius/client.conf:

```
client localhost {
        secret          = lamiachiavesegreta
        shortname       = localhost
}
```

Dobbiamo aggiungere delle voci anche per i nostri access point o switch managed.



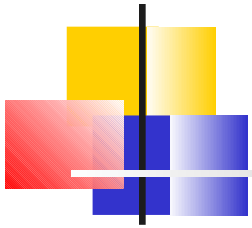
Configurare i client

Supponiamo di avere più dispositivi:

```
client 192.168.1.0/24 {
      secret          = chiavedispositivi
      shortname       = infrastruttura
}
```

Per una maggiore sicurezza conviene creare una definizione per ogni singolo dispositivo.

Rendiamo più sicura l'autenticazione

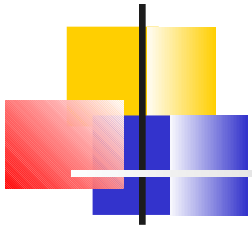


Apriamo il file eap.conf:

```
eap {
      [omissis]
      default_eap_type = peap
      [omissis]
}
```

Con queste righe forziamo i supplicant a tentare per prima l'autenticazione con il protocollo peap.

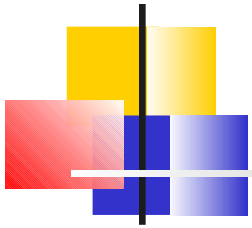
Rendiamo più sicura l'autenticazione



Apriamo il file modules/mschap.conf:

```
mschap {
        use_mppe = yes
        require_encription = yes
        require_strong = yes
}
```

Con queste righe ci assicuriamo che durante l'autenticazione mschap vengano criptate le credenziali di accesso.

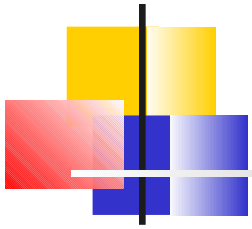


Aggiungiamo gli utenti

Apriamo il file users:

kbyte Cleartext-Password := "miapassword"

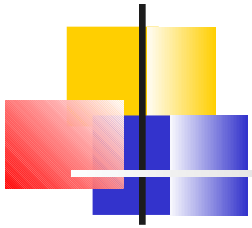
Uno dei difetti di questa soluzione è che le passoword sono in chiaro e in bella mostra sul server. Inoltre la gestione degli utenti è farraginosa e necessita dell'intervento umano.



Altri usi del file users

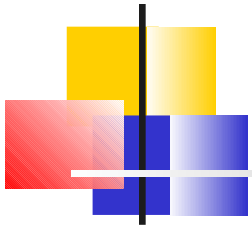
```
lameuser     Auth-Type := Reject
             Reply-Message = "Your account has been disabled."


steve  Cleartext-Password := "testing"
       Service-Type = Framed-User,
       Framed-Protocol = PPP,
       Framed-IP-Address = 172.16.3.33,
       Framed-IP-Netmask = 255.255.255.0,
       Framed-Routing = Broadcast-Listen,
       Framed-Filter-Id = "std.ppp",
       Framed-MTU = 1500,
       Framed-Compression = Van-Jacobsen-TCP-IP
```

Un'autenticazione ancora più sicura: i certificati



A differenza del protocollo PEAP, il protocollo EAP-TLS non usa username e password.

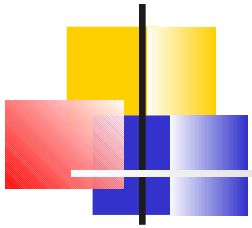
- Non vi sono più username e password da ricordare che possono essere facilmente trafugati.
- Viene usato un certificato SSL
- Il certificato può essere ulteriormente protetto da una password, per evitare il suo utilizzo una volta trafugato.



Preparativi

Anche se il sistema crea dei suoi certificati di default, dobbiamo generarne di nostri:

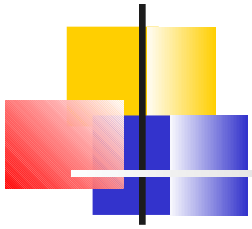
```
cp /usr/share/doc/freeradius/examples/certs/*   \
                                  /etc/freeradius/certs/

cd /etc/freeradius/certs
```



Il file ca.cnf

In questo file dobbiamo modificare le seguenti voci:

```
[ req ]
prompt                 = no
distinguished_name     = certificate_authority
default_bits           = 2048
input_password         = whatever
output_password        = whatever
x509_extensions        = v3_ca

[certificate_authority]
countryName            = FR
stateOrProvinceName    = Radius
localityName           = Somewhere
organizationName       = Example Inc.
emailAddress           = admin@example.com
commonName             = "Example Certificate Authority"
```

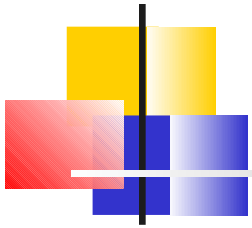


Generiamo il certificato CA

Eseguiamo i seguenti comandi:

```
make ca.pem
make ca.der

touch index.txt
echo 1000 > serial
```

Ora possiamo passare allo step successivo.



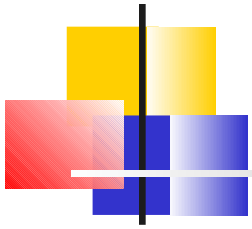
Il file server.cnf

In questo file dobbiamo modificare le seguenti voci:

[req]
prompt = no
distinguished_name = server
default_bits = 2048
input_password = **whatever**
output_password = **whatever**

[server]
countryName = **FR**
stateOrProvinceName = **Radius**
localityName = **Somewhere**
organizationName = **Example Inc.**
emailAddress = **admin@example.com**
commonName = **"Example Server Certificate"**

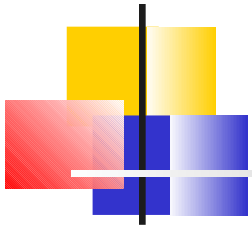
Generiamo il certificato del server



Eseguiamo i seguenti comandi:

make server.pem
make server.csr

Ora possiamo passare allo step successivo.



Il file client.cnf

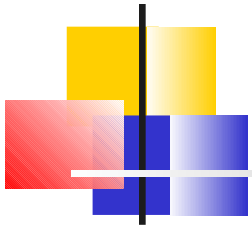
In questo file dobbiamo modificare le seguenti voci:

[req]
prompt = no
distinguished_name = client
default_bits = 2048
input_password = **whatever**
output_password = **whatever**

[client]
countryName = **FR**
stateOrProvinceName = **Radius**
localityName = **Somewhere**
organizationName = **Example Inc.**
emailAddress = **nomeutente@realm.com**
commonName = **nomeutente@realm.com**

Nota: dobbiamo ripetere questa operazione per tutti i client che vogliamo creare.

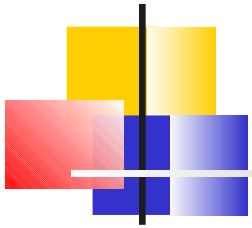
Generiamo il certificato del client



Eseguiamo il seguente comando:

make client.pem

Verrà generato un file .pem con il nome dell'indirizzo email dell'utente che avete specificato. Esso deve essere dato all'utente il quale dovrà utilizzarlo con wpasupplicant o nelle opzioni di rete di windows.

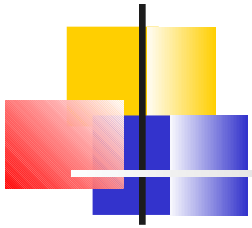


Modifica al file users

Dobbiamo dire a Freeradius di tentare di autenticare gli utenti anche quando non forniscono un UserID.

DEFAULT Auth-Type := EAP

Gli utenti già definiti non sono affetti da questa modifica.



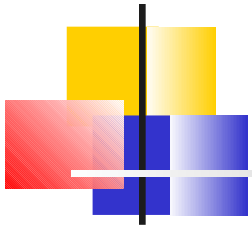
Modifiche al file eap.conf

Dobbiamo dire a freeradius di utilizzare i certificati:

```
tls {
 [omissis]
 # make_cert_command = "${certdir}/bootstrap"
 private_key_password = whatever
 # check_crl = yes ← ma in produzione potrebbe servire
 [omissis]
}
```

Tutte le modifiche avranno effetto al riavvio del servizio Radius.

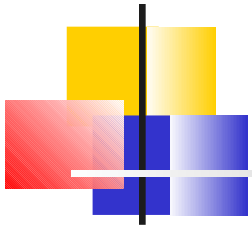
Un piccolo test di funzionamento



Con radtest possiamo vedere se i componenti base del radius funzionano:

radtest -t mschap kbyte miapassword localhost 0 testing123

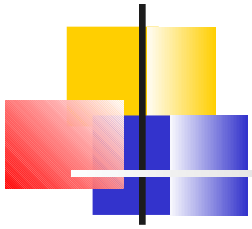
Sending Access-Request of id 201 to 127.0.0.1 port 1812
	User-Name = "kbyte"
	NAS-IP-Address = 67.215.65.132
	NAS-Port = 0
	MS-CHAP-Challenge = 0xb0872e8c7f42dbc7
	MS-CHAP-Response =
0x000100000000000000000000000000000000000000000000970d0
67d284a4636d85dca303db337f81a215b9257002ec5
rad_recv: Access-Accept packet from host 127.0.0.1 port 1812, id=201,
length=84
	MS-CHAP-MPPE-Keys =
0xfda95fbeca288d44ac0782e2de2337dee40e54ee732c1af500000000000
00000
	MS-MPPE-Encryption-Policy = 0x00000002
	MS-MPPE-Encryption-Types = 0x00000004



Il sistema è pronto?

Il sistema realizzato può:

- Essere utilizzato da un sistema di hostspot quale CoovaChilli o similari.
- Autorizzare gli utenti ad accedere alla rete utilizzando un username ed una password
- Autenticare gli utenti ad accedere alla rete utilizzando un certificato ssl.



Contenitori di credenziali

Esiste un modo più efficiente di conservare le credenziali degli utenti?

- **Database relazionale:** freeradius supporta sia Mysql che Postgres nei suoi moduli di base. In questo modo è anche possibile utilizzare interfacce web come DialupAdmin e deloRADIUS.
- **Kerberos:** è a sua volta un protocollo di autenticazione, che può essere utilizzato da freeradius per autenticare gli utenti
- **Server LDAP:** ha un funzionamento analogo ad un database relazionale, ma garantisce migliori prestazioni negli accessi in lettura. Il contro è la difficoltà di gestione.