Gestione dell'Accesso Remoto: RADIUS

Giuseppe Bianchi

22 ottobre 2025

Indice

1 Introduzione

RADIUS (Remote Authentication Dial In User Service) è un protocollo standardizzato in **RFC 2865** (Giugno 2000), progettato per fornire servizi centralizzati di autenticazione, autorizzazione e accounting (AAA). È ampiamente utilizzato nelle reti di accesso remoto e wireless per gestire gli utenti in modo sicuro ed efficiente.

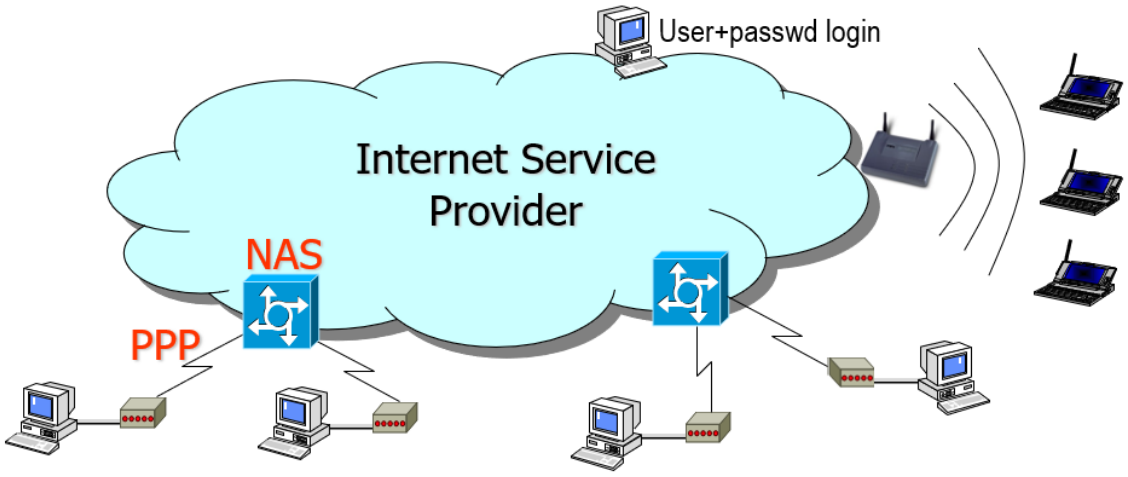
Un'analisi dettagliata della sicurezza di RADIUS è disponibile in Joshua Hill, *"Un'analisi del Protocollo di Autenticazione RADIUS"*.

2 Motivazioni e Contesto

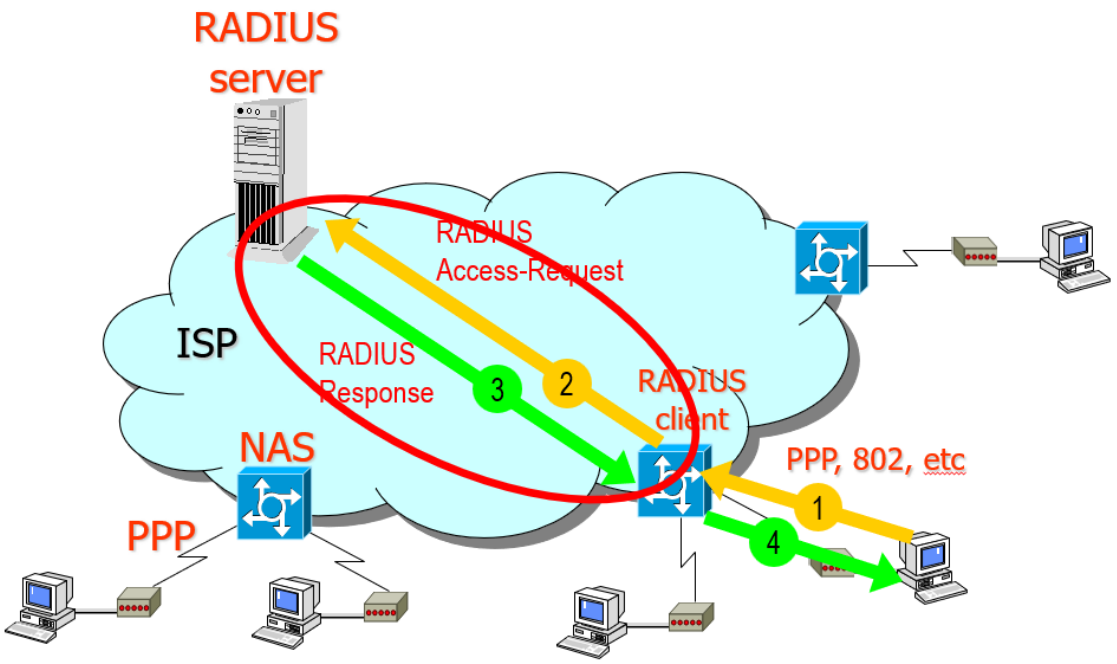
La gestione di reti su larga scala comporta complessità notevoli, specialmente in ambienti con numerosi Network Access Server (NAS) e molteplici servizi di accesso. RADIUS nasce per semplificare questa gestione centralizzando:

- le informazioni sugli utenti,
- le politiche di accesso,
- le configurazioni di servizio.

Un unico database di utenti garantisce coerenza, scalabilità e maggiore sicurezza nella gestione delle credenziali.



3 Posizionamento di RADIUS nella Rete



4 Il modello AAA

RADIUS implementa tre funzioni principali:

Autenticazione: verifica dell'identità dell'utente.

Autorizzazione: controllo dei diritti di accesso.

Accounting: raccolta di dati relativi all'utilizzo (es. byte trasmessi, tempo di connessione).

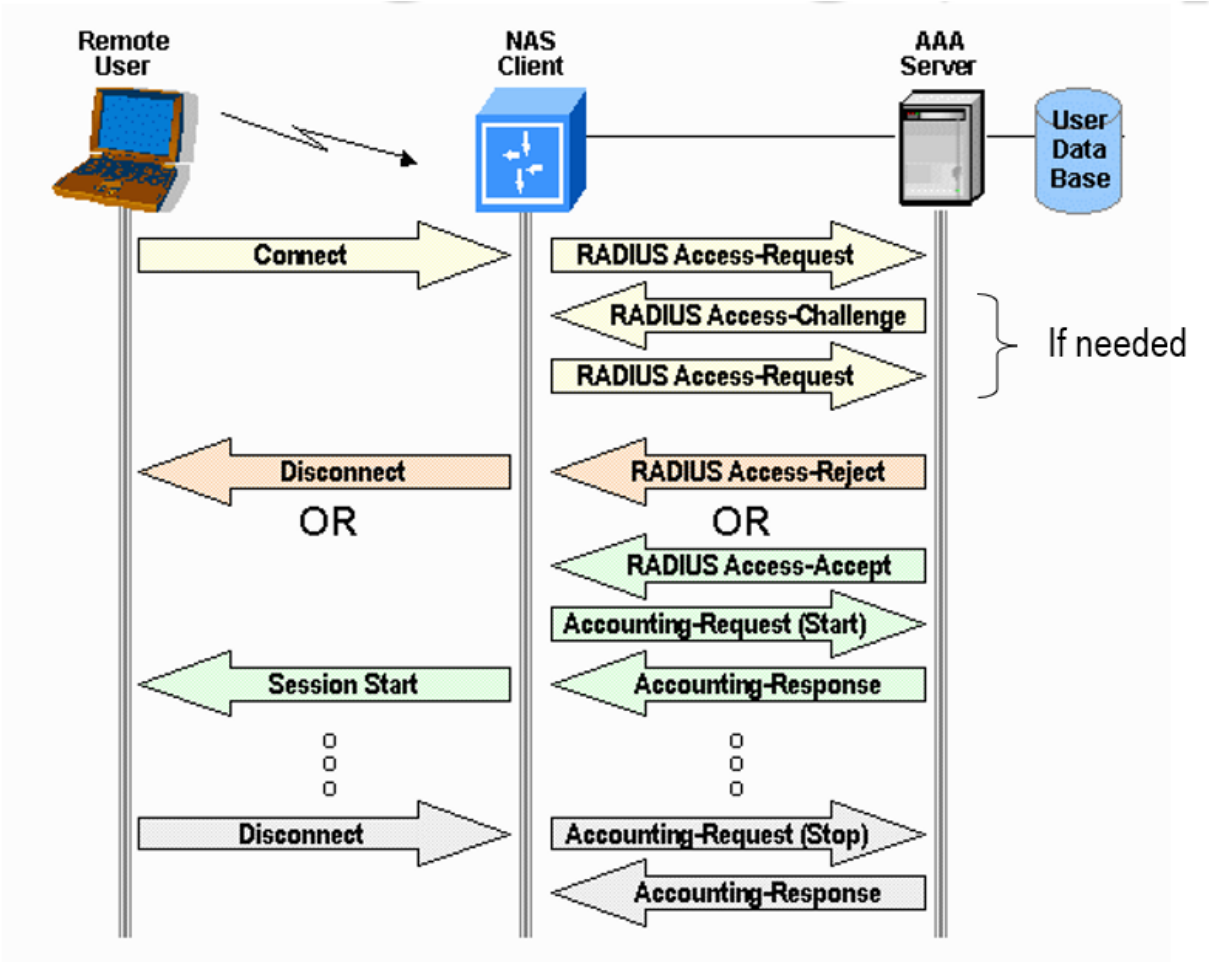
5 Architettura Client-Server

RADIUS è un protocollo **client-server** basato su **UDP/IP**:

- **Client RADIUS:** il NAS (non l'utente finale).
- **Server RADIUS:** risponde alle richieste di autenticazione/autorizzazione.

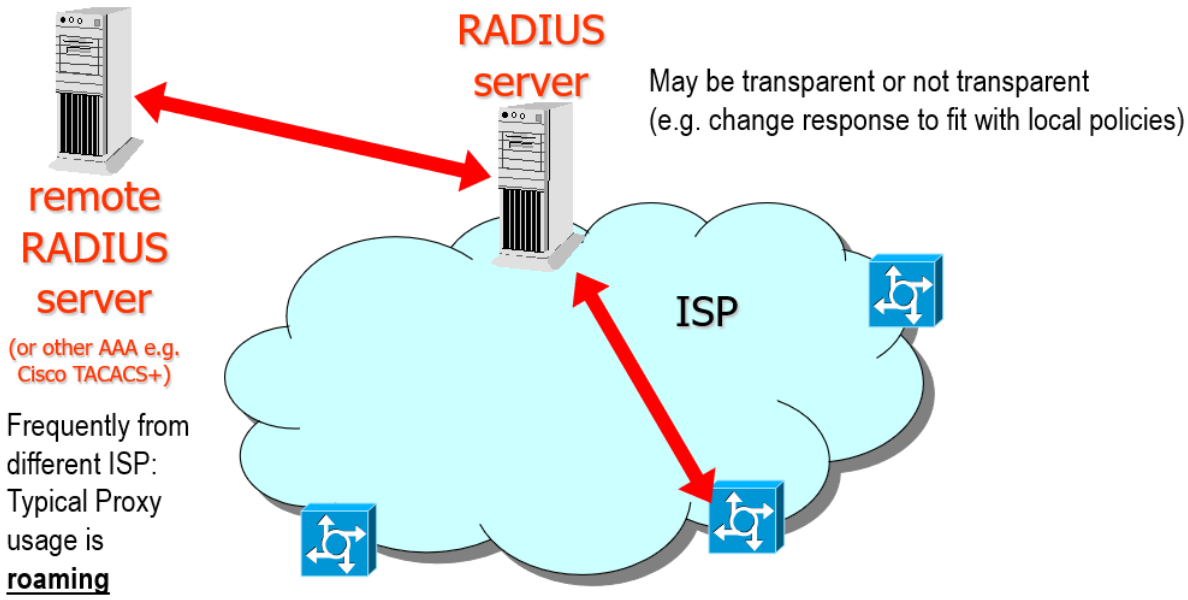
Le porte standard sono:

- 1812 per l'autenticazione,
- 1813 per l'accounting.



5.1 Architettura Proxy

Il server RADIUS può agire come **proxy**, inoltrando richieste verso altri server:



Questo è tipico in scenari di **roaming** tra ISP differenti.

6 Architettura del Server RADIUS

Il server gestisce vari database:

- **Utenti:** credenziali, metodi di autenticazione, attributi di autorizzazione.
- **Client:** NAS autorizzati a comunicare con il server.
- **Contabilità:** dati relativi all'uso della rete.

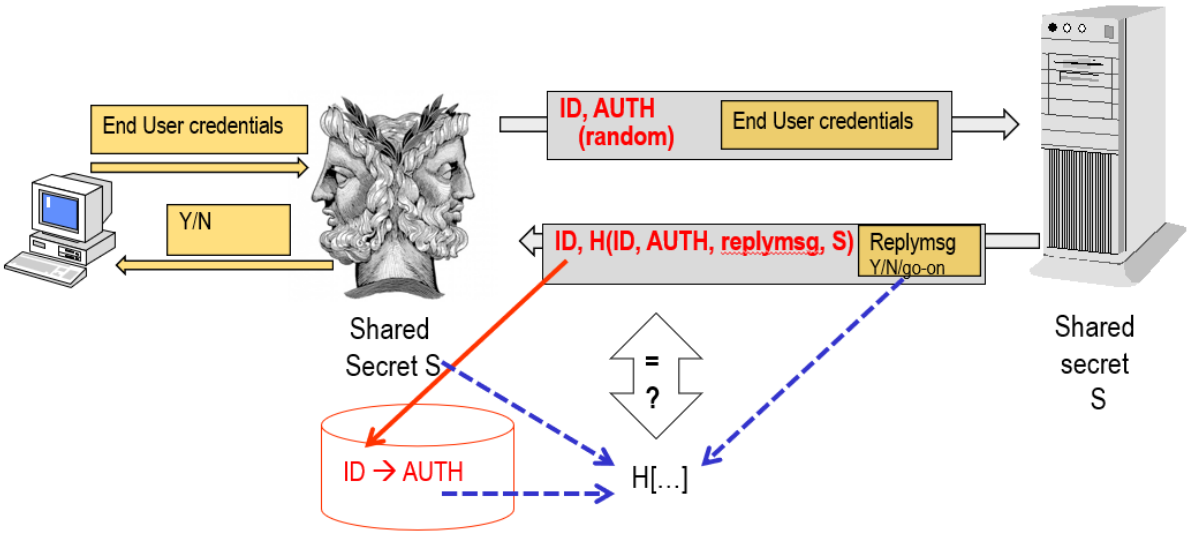
7 Sicurezza di RADIUS

7.1 Caratteristiche principali

- Autenticazione della risposta tramite segreto condiviso.
- Password utente crittografata, mai trasmessa in chiaro.

7.2 Limitazioni

- Solo la risposta è autenticata.
- Uso di **MD5** non HMAC.
- Chiave condivisa spesso a bassa entropia.

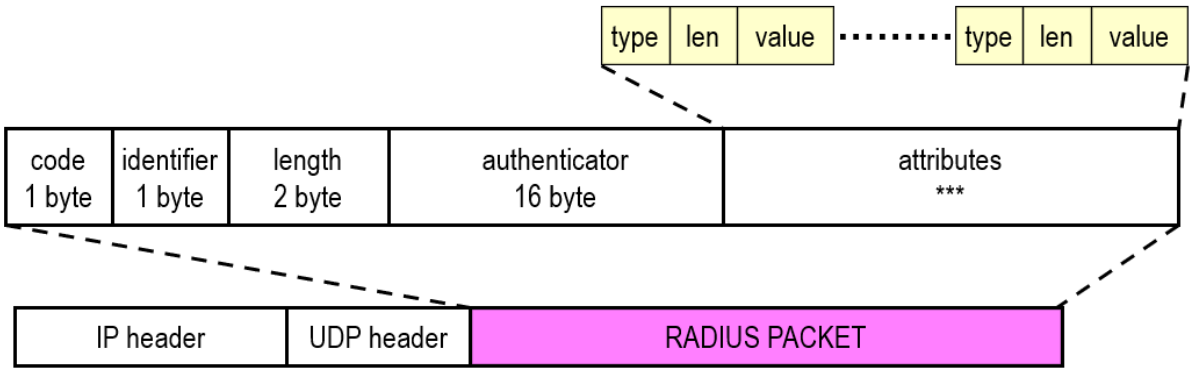


8 Formato del pacchetto RADIUS

Ogni pacchetto è composto da:

- **Codice** (1 byte)
- **Identificatore** (1 byte)
- **Lunghezza** (2 byte)
- **Authenticator** (16 byte)
- **Attributi** (variabili)

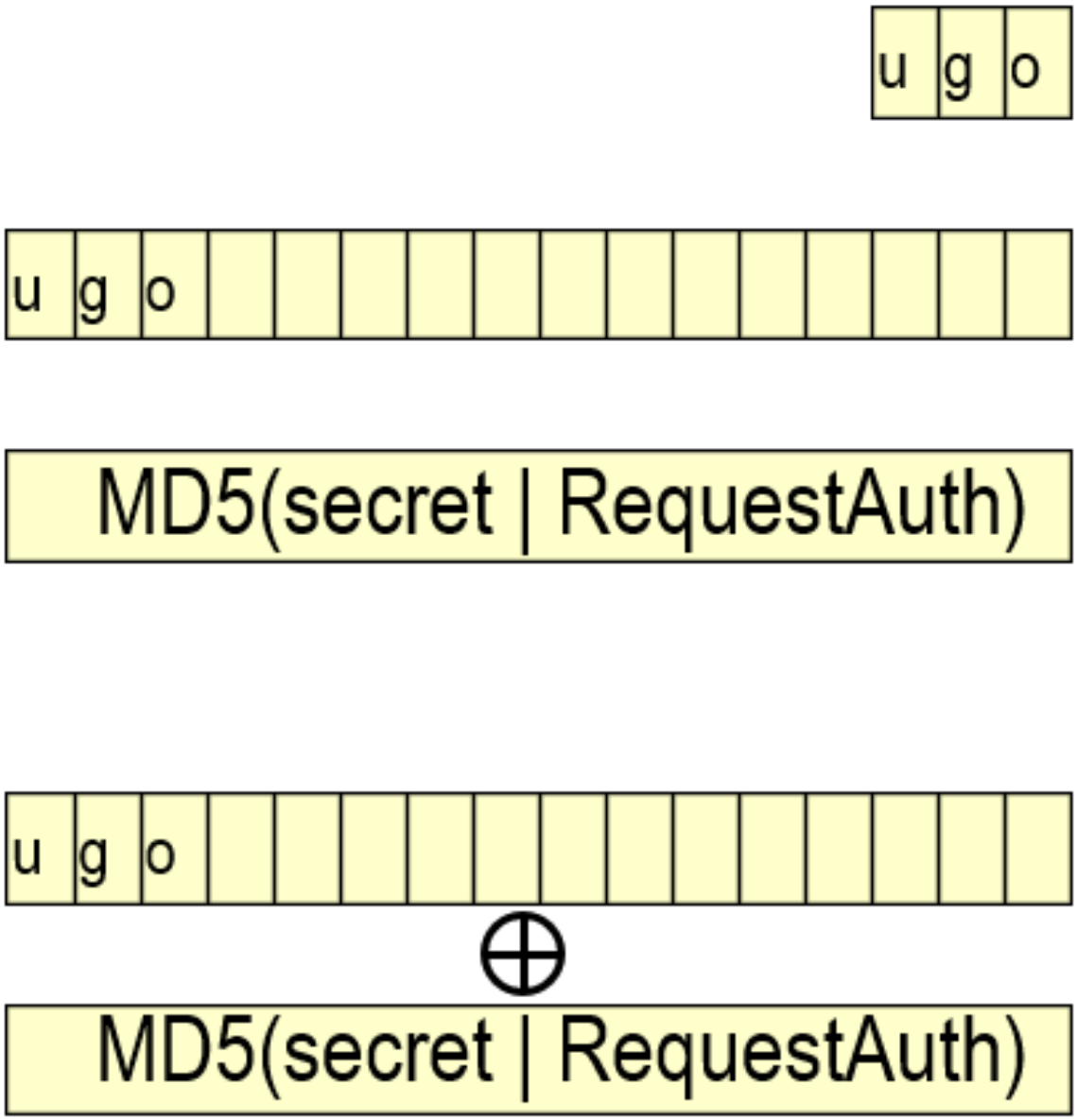
Code (dec)	Packet
1	Access-Request
2	Access-Accept
3	Access-Reject
4	Accounting-Request
5	Accounting-Response
11	Access-Challenge



9 Crittografia della Password

Il meccanismo di cifratura della password utente è basato su:

1. Padding della password a multipli di 16 byte.
2. Calcolo di `MD5(segreto | RequestAuth)`.
3. XOR tra password e hash.
4. Iterazione per segmenti >16 byte.



10 Messaggi Principali

10.1 Access-Request

Contiene:

- User-Name
- User-Password o CHAP-Password
- NAS-IP-Address

- NAS-Port

10.2 Access-Accept

Conferma positiva dell'autenticazione e specifica il tipo di servizio (es. PPP, login, callback).

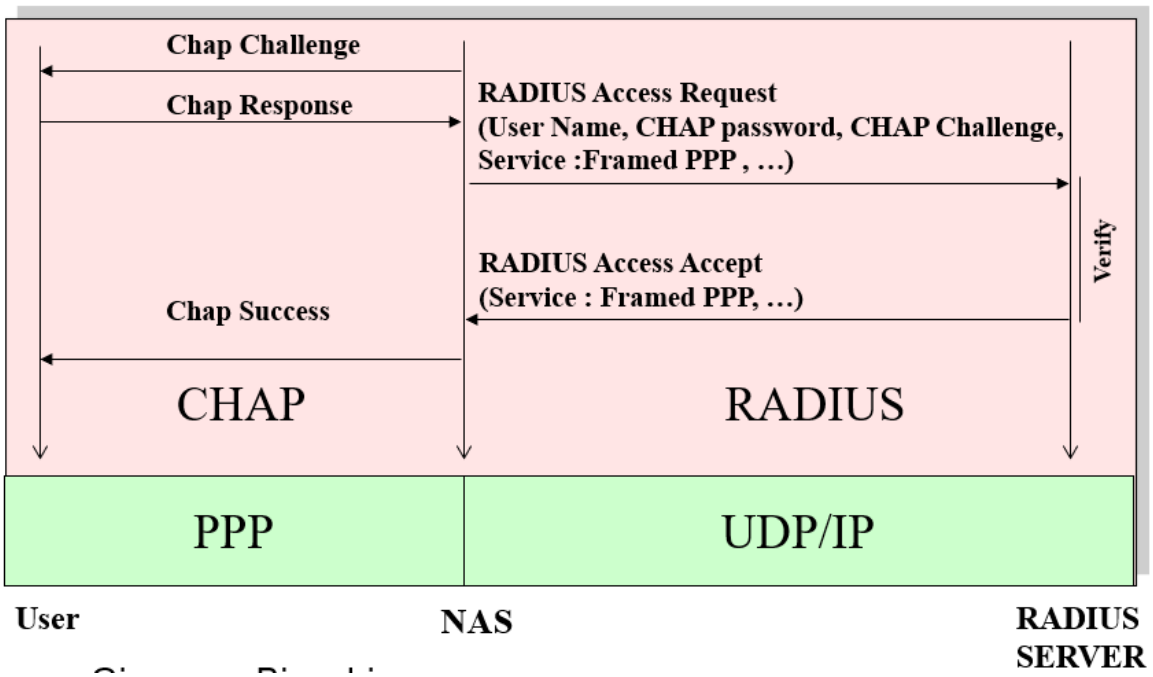
10.3 Access-Reject

Indica fallimento dell'autenticazione o mancata autorizzazione.

10.4 Access-Challenge

Richiede ulteriori informazioni dall'utente, tipicamente in autenticazioni a più fasi (es. CHAP).

- Usato quando il server vuole/ha bisogno che l'utente invii una risposta ulteriore
  - Ad esempio, meccanismi di autenticazione challenge/response
    - Non necessariamente CHAP (vedi supporto CHAP più avanti)! Potrebbe essere supporto RADIUS per autenticazione GSM/UMTS!
  - Ad esempio, chiedere all'utente di inserire una password
- La challenge tipicamente contiene
  - Uno o più attributi reply-message
    - Che possono essere usati in modo molto flessibile
  - Può contenere testo da mostrare all'utente
  - Può contenere una challenge esplicita di autenticazione
- Il NAS raccoglie la risposta dall'utente e invia una NUOVA Access-Request
  - Nuovo ID
  - Nuova User-Password - contiene la risposta dell'utente (criptata)
- In base a questo, il server accetta, rifiuta o invia un'altra challenge



11 Sicurezza e Vulnerabilità

11.1 Problemi principali

- Dati in chiaro (User-Name, NAS-ID).
- Access-Request non autenticato.
- Dipendenza da MD5.

11.2 Message-Authenticator

Estensione (tipo 80) che introduce un campo HMAC-MD5 sull'intero pacchetto, rendendolo obbligatorio con EAP (RFC 2869).

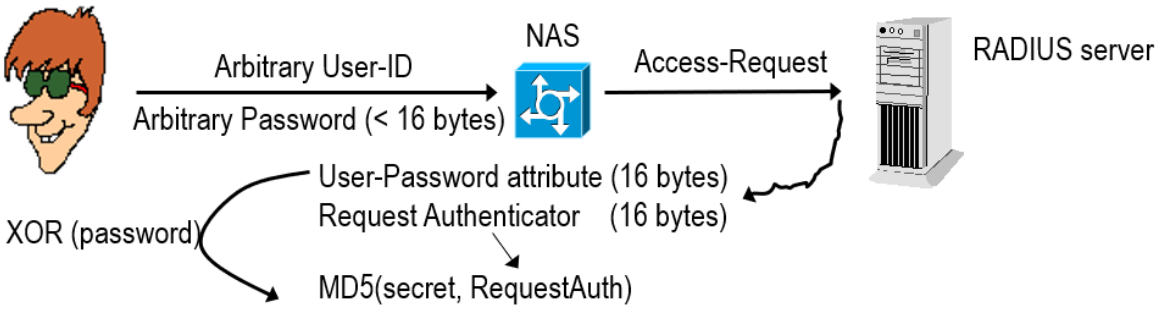
- Firma del messaggio
  - Principalmente per i messaggi Access-Request
    - Poiché non sono autenticati
    - Naturalmente può essere usata anche nei pacchetti Reject/Accept/Challenge
  - DEVE essere usata quando EAP è usato con RADIUS (RFC 2869)
    - Può naturalmente essere usata con altri metodi di autenticazione
- Sintassi tipica degli attributi

Tipo=80	Len=18	Autenticatore (16 byte)

- Autenticatore =
  - HMAC-MD5(intero pacchetto) =
  - HMAC-MD5(tipo | ID | len | RichiestaAuth | attributi)
- Nel calcolo, Autenticatore = 0000.0000.0000.0000
- Chiave condivisa usata come chiave per la funzione hash HMAC-MD5

11.3 Attacchi noti

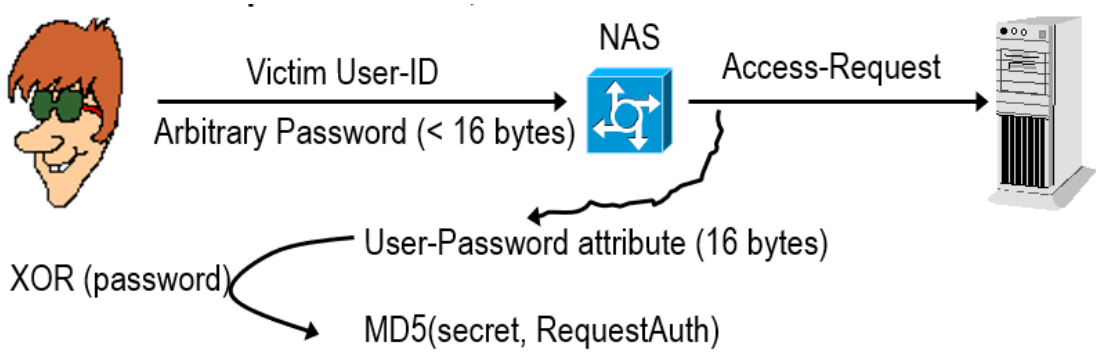
- **Dizionario sul segreto condiviso:** possibile se segreto a bassa entropia.
  - Di solito segreto condiviso a bassa entropia
    - Molte implementazioni permettono solo segreti condivisi ASCII
  - E spesso ne viene usato uno solo per tutta la rete!!
    - Storia curiosa: hack del Fonera! `http://stefans.datenbruch.de/lafonera/`
    - Lezione: usare un segreto condiviso per ogni client (identificato tramite indirizzo IP del client)
  - Molti agganci per attacco dizionario offline:
    - Intercetta qualunque coppia richiesta-risposta
      - La richiesta ti dà un ReqAuth casuale
      - RespAuth = MD5(Codice | ID | Lunghezza | ReqAuth | Attributi | Segreto)
      - Segreto inserito alla fine: la precomputazione MD5 rende l'attacco più facile!
    - Non serve neanche ottenere una coppia, basta una richiesta con la password utente!
      - Idea terribile: riutilizzare lo stesso segreto anche per la cifratura delle password!!



- **Password cracking:** sfruttando autenticazioni ripetute o PRNG deboli.

  PRIMO PASSO: come il caso precedente, ma con un ID utente valido:

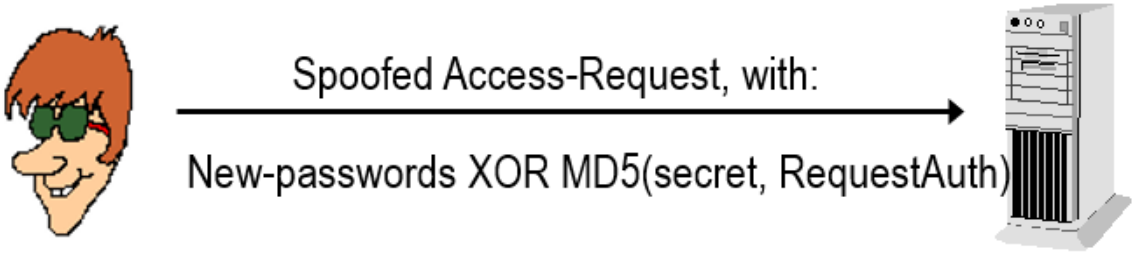
  - Victim User-ID
  - Password arbitraria (< 16 byte)
  - NAS invia Access-Request al server RADIUS
  - Attributo User-Password (16 byte) calcolato come:
    - XOR(password) con MD5(segret, RequestAuth)



SECONDO PASSO: L'attaccante ora è in grado di "cifrare" la password utente! Può sfruttare:

1. Assenza di limite superiore sul rate di autenticazione lato server (i limiti imposti ai client vengono aggirati)
2. I server RADIUS tipicamente non controllano il riuso dell'autenticatore

Funziona solo con password di 16 byte o meno (nella maggior parte dei casi).



12 PRNG

12.1 Implementations

- Sicurezza di radius:
  - Richiede l'unicità del Request Authenticator
    - Deve essere un nonce!
- Alcune implementazioni:
  - Poveri Generatoridi Numeri Pseudo-Casuali (PRNG)
    - Spesso generatori non criptografici: terribile!!
    - Cicli corti, prevedibilità, ...
- Tre verifiche (quando si affronta un PRNG debole)
  - Qual è il «ciclo»?
  - E la prevedibilità?
    - Esempio: LCG, se si conosce un valore si conoscono tutti!!
  - Valori unici o ripetuti?
    - Mersenne Twister: ciclo di $2^{19937} - 1$, ma i valori si ripetono

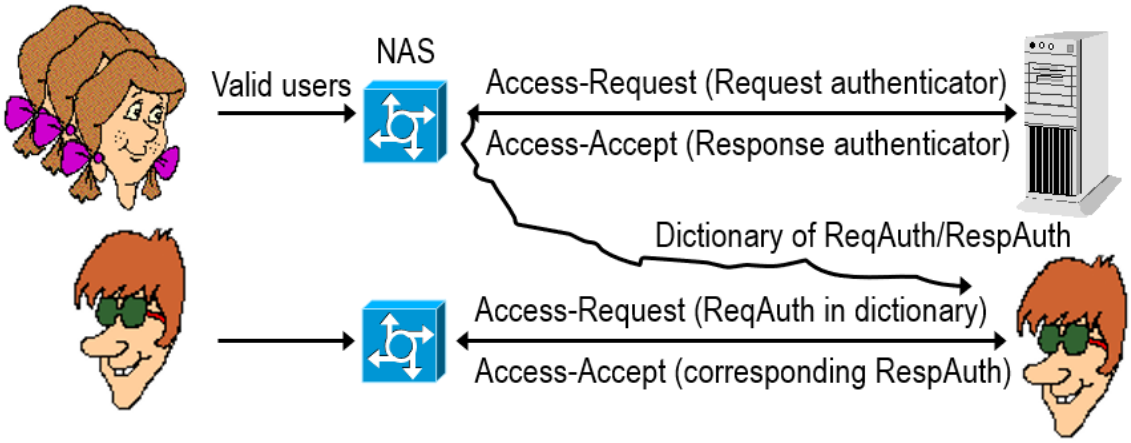
12.2 note

- Assumiamo ciclo PRNG = 2^N
  - Al massimo solo 2^N valori diversi!
  - Attacco del compleanno: probabilità 1/2 con circa $2^{N/2}$
  - $N = 16$ attacco del compleanno con circa $o(250)$ pacchetti!
  - C drand48 $\rightarrow$ attacco del compleanno con circa $o(16M)$ pacchetti
    - (non sono davvero tanti!)
- Idea di implementazione "terrificante (?)"
  - Si parte da un numero casuale a 32 bit

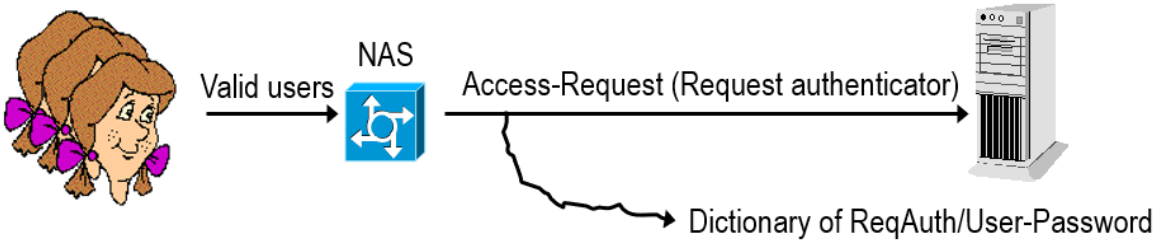
- Si genera un autenticatore a 128 bit concatenando 4 numeri casuali consecutivi da 32 bit
- Ciclo: ridotto a 30 bit!!
    - Attacco del compleanno: circa $o(32000)$ pacchetti!

12.3 Replay attacks

autenticare/autorizzare un utente illegale senza password valida



Passivamente monitorare il traffico di rete permette di costruire un dizionario degli Authenticator di richiesta e degli attributi User-Password (protetti) associati.

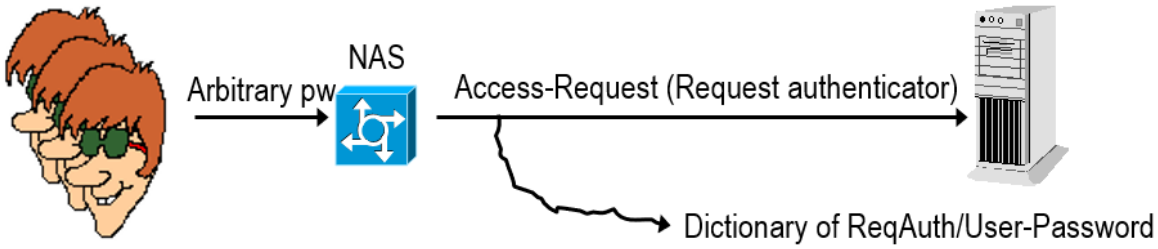


- Authenticator di richiesta ripetuto osservato
- XOR della precedente user-password con una nuova user-password
    - Da utenti diversi
- Risultato: poiché ReqAuth è lo stesso
    - (user-password #1) XOR (user-Password #2) =
    - = [pw_user1 XOR MD5(segreto,ReqAuth)] XOR [pw_user2 XOR MD5(segreto,ReqAuth)] =

  - = pw_user1 XOR pw_user2

- MA le password di utenti diversi differiscono in lunghezza:
  - Gli ultimi caratteri della password più lunga sono in chiaro!
  - Le dimensioni delle password sono note!!
  - Le password a bassa entropia aiutano a divulgarle!

Attivamente inviare password scelte dall'attaccante per aggiungere password note al dizionario degli Authenticator di richiesta e degli attributi User-Password (protetti) associati.



- **Non serve indovinare!**
- Il dizionario è costruito usando i tentativi rifiutati!

13 Lezioni Apprese

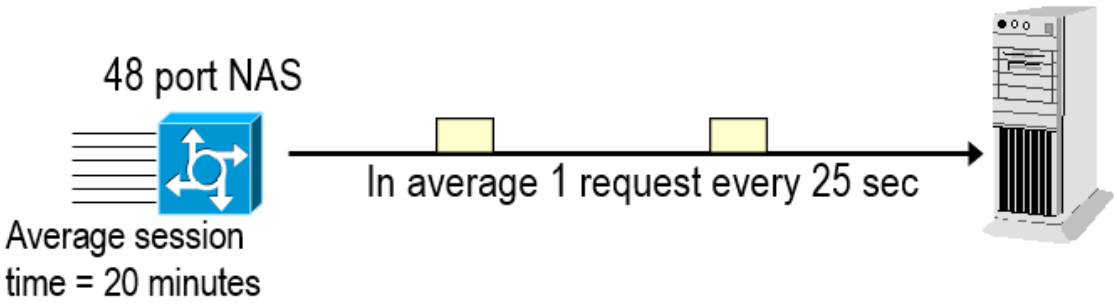
- I protocolli AAA devono appoggiarsi a livelli di sicurezza sottostanti (es. IPsec, TLS).
- MD5 non è più sicuro dal 2005.
- Migrazione verso protocolli moderni come Diameter o RADSEC.

14 Radius: Oggi

- Radius inizialmente distribuito principalmente per supportare utenti dial-up in PPP e utenti terminali di login
- Oggi, RADIUS = standard de facto per AAA
  - Supporto universale dai fornitori di dispositivi
- Ma gravi limiti funzionali

- Scalabilità: oggi la base clienti è molto più ampia
  - Da poche migliaia di utenti, come nei primi ISP
  - A diversi milioni di utenti come negli operatori cellulari e ISP a livello nazionale
- Estensibilità:
  - Nuove tecnologie
    - Wireless, DSL, 3G/4G/5G, Ethernet, ...
  - Molti più scenari di servizio
    - Mobile-IP, roaming, carrier-grade accounting, ecc.
    - Nuovi meccanismi di autenticazione
- Interoperabilità: molti aspetti lasciati non specificati e gestiti in modo proprietario
  - Esempi: failover, bilanciamento del carico, selezione del server, ecc.

14.1 Scalabilità



- Cosa succede se ci sono 10.000 di questi NAS?
  - 400 richieste di accesso al secondo
    - 3.2 Mbps di carico sul server se la richiesta è in media 1KB
  - Aggiungi 400 richieste di accounting al secondo
  - Aggiungi la consegna dei record di accounting
- Alcune implementazioni RADIUS potrebbero non essere in grado di gestire tutto questo carico senza perdita di pacchetti!
- Infine, aggiungi picchi di carico

  - Quando il NAS si riavvia (ad es. dopo un blackout)
    * Molti record di accounting sono inviati uno dopo l'altro
    * Gli utenti remoti eseguono il login contemporaneamente

Conclusione: i server AAA (e le loro reti) possono subire CONGESTIONE!! Così come perdita di pacchetti!!

Standardizzazione IETF

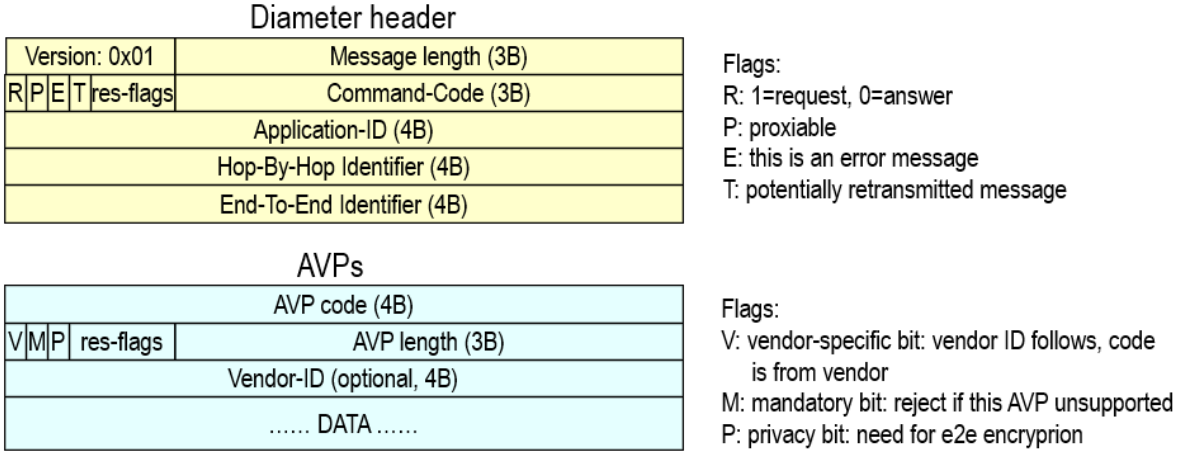
- **Diameter**
  - Avviato a dicembre 1998
  - Ora completato, le attività si sono trasferite su DiME (Diameter Maintenance and Extensions WG)
    * Prima email del WG a gennaio 2006
    * Azioni pianificate: supporto per MIPv6; QoS; AAA in telefonia IP (SIP) e nelle reti locali LAN (VLAN)
- **RADext**
  - Avviato ad agosto 2003
  - Estensioni RADIUS con compatibilità all'indietro obbligatoria
    * Azioni pianificate: AAA in telefonia IP (SIP) e nelle reti locali LAN (VLAN), supporto pre-pagato
  - Nessun trasporto (ancora UDP) e miglioramenti di sicurezza
- **Compatibilità RADIUS/Diameter**
  - Obiettivo di entrambi i WG!!

Perché "duplicare" il lavoro?

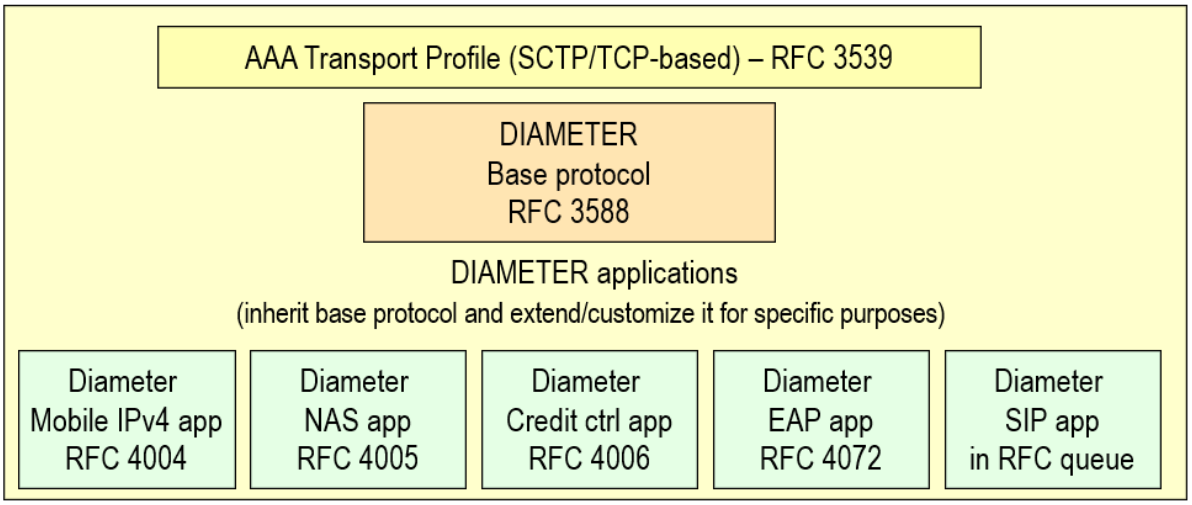
- **RADIUS:**
  - Enorme lavoro sul deployment di RADIUS
    * Standard de facto, fortemente integrato nei domini aziendali/ISP

  - Le limitazioni possono essere aggirate con “piccole” estensioni ad hoc

- **Diameter:**
  - Protocollo completamente nuovo
    - Sebbene in parte compatibile all’indietro, cambia comunque molti concetti base
      - Trasporto, formato del pacchetto, filosofia, ...
  - Molto più potente
    - Ma anche molto, molto più complesso
  - Scelta ideale in nuovi scenari (emergenti)
    - Specialmente nel mondo della mobilità 3G
- **Incertezza su quale sarà IL futuro protocollo AAA e se ci sarà UN protocollo dominante: tenere entrambi!**
- Estensione dei limiti funzionali
  - Campo AVP a 24 bit (Attribute-Value-Pair) contro il campo attributo a 8 bit (esaurito)
  - Niente più ID a 8 bit (limite di 256 pacchetti in volo dal client allo stesso server – non scalabile), ma ora: 32 bit (identificatore Hop-by-Hop)
  - Rilevamento dei duplicati (identificatore E2E e flag T)
  - Negoziazione delle capacità: AVP obbligatori/non obbligatori (flag M) permettono di limitare il rifiuto solo a gravi mancanze di capacità
    - Versus risposta “reject” di radius se gli attributi richiesti non sono supportati



15 Evoluzione: Diameter



15.1 Motivazioni

Diameter nasce per superare le limitazioni di RADIUS:

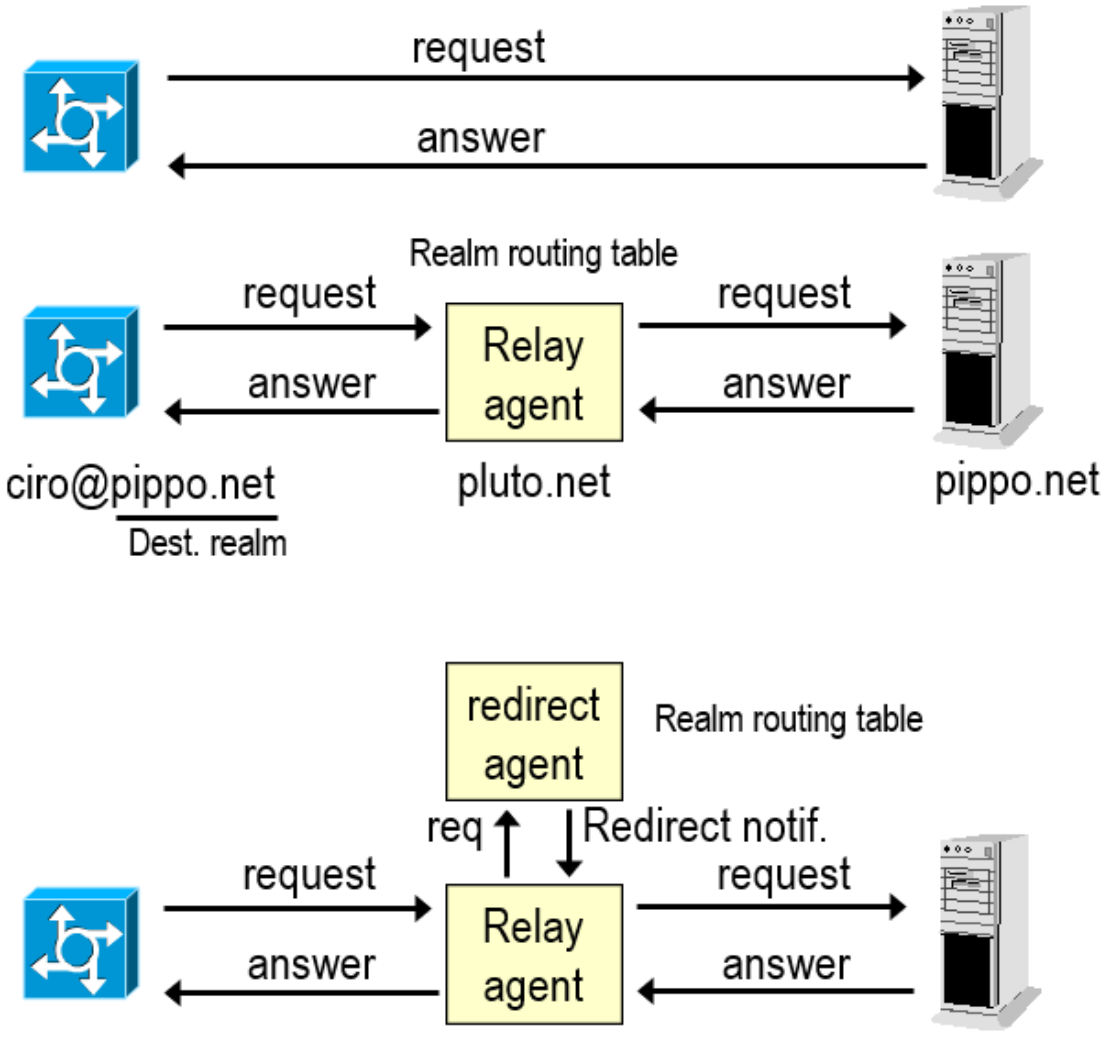
- Trasporto affidabile
  - Versus UDP poco affidabile di RADIUS e ritrasmissione "proprietaria"
- Affidabilità essenziale nell'accounting
  - Perdita di pacchetti = perdita di denaro!
- SCTP preferito (altrimenti TCP)
  - Un'unica connessione persistente tra client e server
  - Pacchetti messi in pipeline in questa singola connessione
- Controllo degli errori e failover standardizzati
  - Versus approcci dipendenti dalla implementazione RADIUS
- Controllo degli errori e ritrasmissioni a livello applicativo
- Rilevamento dei duplicati
  - Particolarmente importante per l'accounting
- Watchdog a livello applicativo: pacchetti periodici per capire quando il peer Diameter fallisce

- Default: 1 messaggio ogni 30 +/- 2 secondi (jitter casuale artificiale aggiunto per evitare sincronizzazione)
- Può essere ridotto a 6 +/- 2 secondi
- Timer azzerato quando arrivano altri pacchetti (il watchdog è usato solo quando nessun altro pacchetto arriva)

15.2 Caratteristiche chiave

- Campo AVP a 24 bit (vs. 8 bit di RADIUS)
- Meccanismi di discovery, proxy e redirect
- Supporto nativo per roaming e messaggi asincroni

16 Agenti



Nessun agente intermedio

Relay agent: accetta la richiesta e la instrada verso il server corretto in base alle informazioni contenute nel messaggio (realm di destinazione, ad esempio ciro@pippo.net)

Proxy agent: come relay ma modifica il messaggio (di conseguenza l'autenticazione end-to-end del messaggio viene compromessa)

Redirect agent: fornisce la decisione di instradamento sulla richiesta in arrivo ma non effettua l'instradamento (restituisce solamente la notifica di redirect) Utile quando le decisioni di instradamento Diameter sono centralizzate (es. per un consorzio di realm). Caso tipico: ciascun relay individuale instrada di default verso il redirect agent.

17 Conclusioni

RADIUS, pur con i suoi limiti, rimane uno standard de-facto grazie alla sua diffusione e interoperabilità. Tuttavia, l'aumento delle esigenze di sicurezza e scalabilità ha portato alla nascita di **Diameter**, progettato per ambienti di rete moderni (3G, 4G, 5G). Entrambi i protocolli continueranno a coesistere, con RADIUS ancora dominante nel breve termine e Diameter sempre più rilevante nei sistemi mobili e ISP di nuova generazione.