Gestione dell'Accesso Remoto: RADIUS

Giuseppe Bianchi

7 dicembre 2025

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

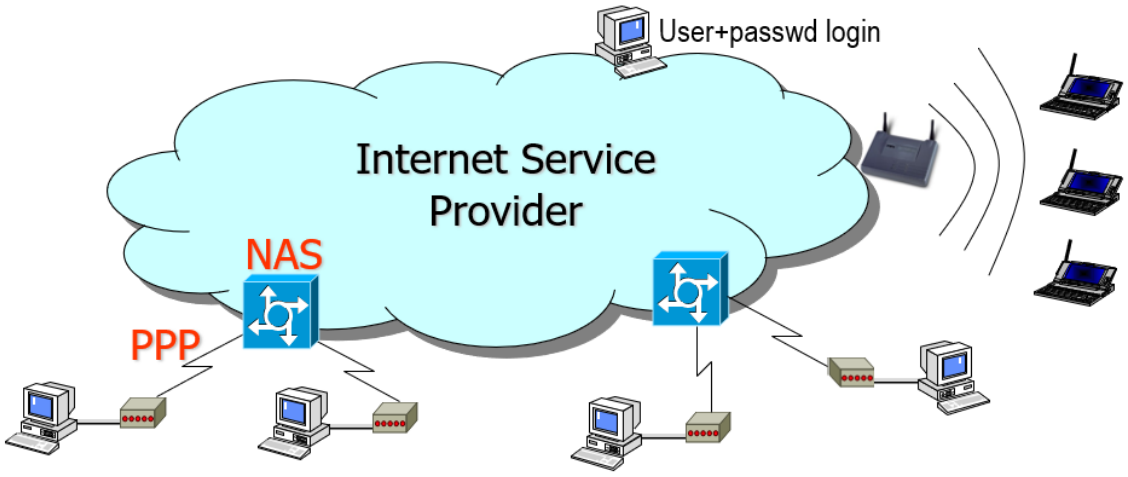
La gestione di reti su larga scala (come quelle degli Internet Service Provider - ISP) comporta complessità notevoli, che possono trasformarsi in un vero e proprio incubo gestionale ("management nightmare").

In uno scenario privo di centralizzazione, ci si scontra con diverse problematiche critiche:

- **Molteplicità dei punti di accesso:** Una rete è composta da numerosi NAS (*Network Access Server*), come router, access point wireless e concentratori VPN. Gestire localmente le credenziali su ogni singolo dispositivo è impraticabile.
- **Eterogeneità dei servizi:** Non si tratta solo di gestire l'accesso base, ma di supportare diversi metodi (Dial-up, Wireless, DSL) e protocolli (PPP, 802.1X).
- **Configurazioni specifiche per servizio:** Oltre alla semplice autenticazione (verificare chi è l'utente), è necessario assegnare configurazioni specifiche al momento della connessione (es. indirizzo IP statico, larghezza di banda, filtri di sicurezza, VLAN).

RADIUS nasce per risolvere questi problemi attraverso la gestione di un **singolo database utenti centralizzato**. Questo approccio garantisce:

- **Scalabilità:** Aggiungere un nuovo utente o modificare una password richiede un solo intervento sul server centrale, non su centinaia di NAS.
- **Coerenza:** Le policy di accesso sono uniformi su tutta la rete.
- **Sicurezza:** Le credenziali sono stoccate in un unico punto blindato e non replicate su dispositivi periferici potenzialmente vulnerabili.



3 Posizionamento di RADIUS nella Rete

Per comprendere il funzionamento di RADIUS, è fondamentale analizzare il flusso di comunicazione e i ruoli dei dispositivi coinvolti. Il protocollo opera secondo un'architettura **Client-Server**, ma con una distinzione importante rispetto alla percezione dell'utente finale.

3.1 Attori coinvolti

Utente Finale: Il dispositivo (PC, Laptop) che richiede accesso alla rete. **Nota bene:** L'utente NON parla direttamente il protocollo RADIUS.

NAS (RADIUS Client): Il dispositivo di rete (Router, Switch, Access Point) che funge da intermediario. È lui il "Cliente" RADIUS che interroga il server.

RADIUS Server: Il server centrale che detiene il database (o si collega ad esso, es. SQL/LDAP) e prende le decisioni di autenticazione.

3.2 Il Flusso di Autenticazione (Step-by-Step)

Come illustrato nello schema sottostante, il processo si divide in quattro fasi distinte, evidenziando dove il protocollo RADIUS è presente e dove no:

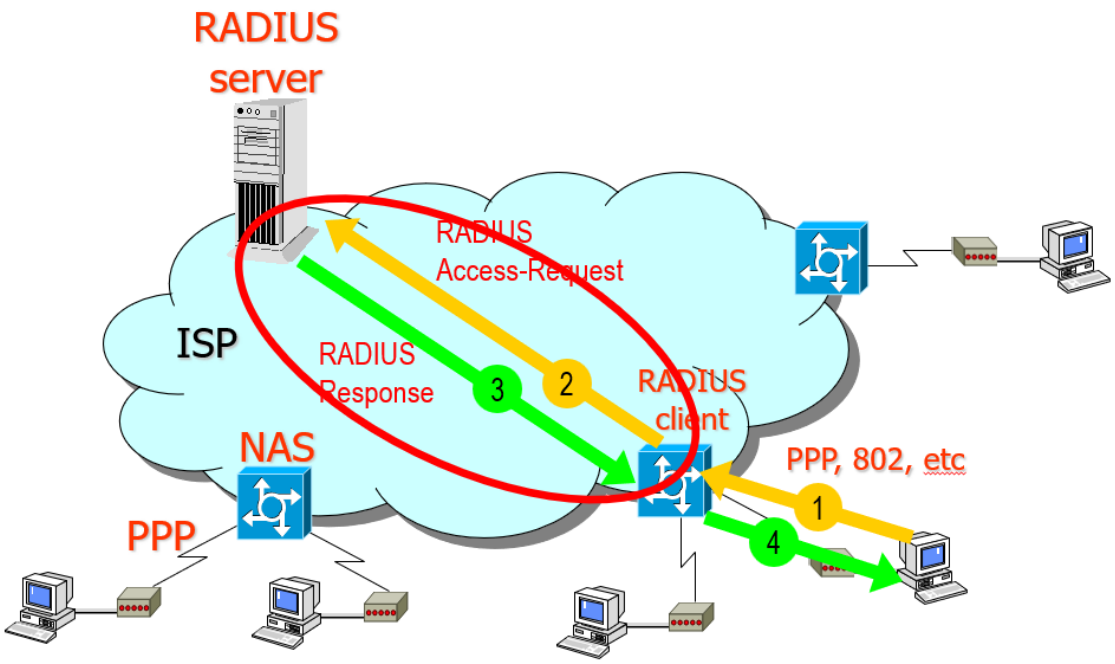
1. **Richiesta Utente → NAS (No RADIUS):** L'utente invia le proprie credenziali al NAS utilizzando protocolli di livello 2 o specifici per il mezzo trasmissivo (es. PPP, EAP su 802.1X, PAP/CHAP). In questa fase, il traffico è locale tra dispositivo e punto di accesso.

2. **NAS → RADIUS Server (RADIUS Access-Request):** Il NAS riceve le credenziali, le "impacchetta" (wrap) all'interno di un pacchetto RADIUS chiamato `Access-Request` e le invia al server. Qui il NAS agisce come client RADIUS.

3. **RADIUS Server → NAS (RADIUS Response):** Il server verifica le credenziali nel database e controlla le policy. Invia una risposta al NAS che può essere:

   - `Access-Accept`: L'utente è autorizzato. Il pacchetto include anche parametri di configurazione (es. IP da assegnare, VLAN ID).
   - `Access-Reject`: L'accesso è negato.
   - `Access-Challenge`: Richiesta di ulteriori informazioni (usato in metodi di autenticazione complessi).

4. **NAS → Utente (No RADIUS):** Il NAS riceve il verdetto dal server. Se è positivo, apre la porta/connessione e notifica l'utente del successo. Se negativo, termina la connessione. Anche qui, l'utente vede solo il risultato della connessione PPP/Network, rimanendo ignaro dell'esistenza del server RADIUS alle spalle.



4 Il modello AAA

Il protocollo RADIUS fornisce funzionalità centralizzate basate sul framework **AAA**. Questo modello scompone la gestione dell'accesso alla rete in tre processi distinti e sequenziali:

Autenticazione (Authentication): È il processo fondamentale di verifica dell'identità. Il sistema pone la domanda: *"Sei davvero chi dici di essere?"*. In questa fase, il server controlla le credenziali fornite dall'utente (come username e password) per assicurarsi che la richiesta provenga da un soggetto legittimo.

Autorizzazione (Authorization): Successiva all'autenticazione, questa fase definisce i permessi. La domanda chiave è: *"Hai i permessi necessari per accedere a questo servizio?"*. Non basta essere un utente valido; il sistema deve determinare quali specifiche risorse o servizi di rete l'utente è abilitato ad utilizzare (es. assegnazione di un indirizzo IP specifico, filtri di sicurezza, o accesso a determinate VLAN).

Accounting (Contabilizzazione): È la fase di tracciamento e monitoraggio. Risponde alla domanda: *"Cosa stai facendo, usando o pagando in questo momento?"*. Il server RADIUS raccoglie dati statistici sull'attività dell'utente, fondamentali per:

- Monitorare il consumo di risorse (es. byte trasmessi e ricevuti).
- Calcolare la durata della connessione.
- Gestire la fatturazione (*billing*) basata sull'effettivo utilizzo della rete.

5 Architettura Client-Server

RADIUS è definito come un protocollo **Client-Server**, ma è fondamentale comprendere chi ricopre questi ruoli per non fare confusione con l'utente finale:

- **RADIUS Client (il NAS):** Il cliente *non* è il PC o lo smartphone dell'utente. Il client è il **NAS** (Network Access Server), ovvero il dispositivo di rete (switch, access point, server VPN) che deve decidere se far passare o meno il traffico.
- **RADIUS Server:** È l'entità centrale (o un cluster di server replicati per ridondanza) che possiede il database e la logica decisionale.

Dal punto di vista del trasporto, il protocollo utilizza **UDP/IP** (User Datagram Protocol). La scelta di UDP (stateless) rispetto a TCP è dovuta alla necessità di efficienza e velocità nelle transazioni di autenticazione.

Le porte standard assegnate dallo IANA sono:

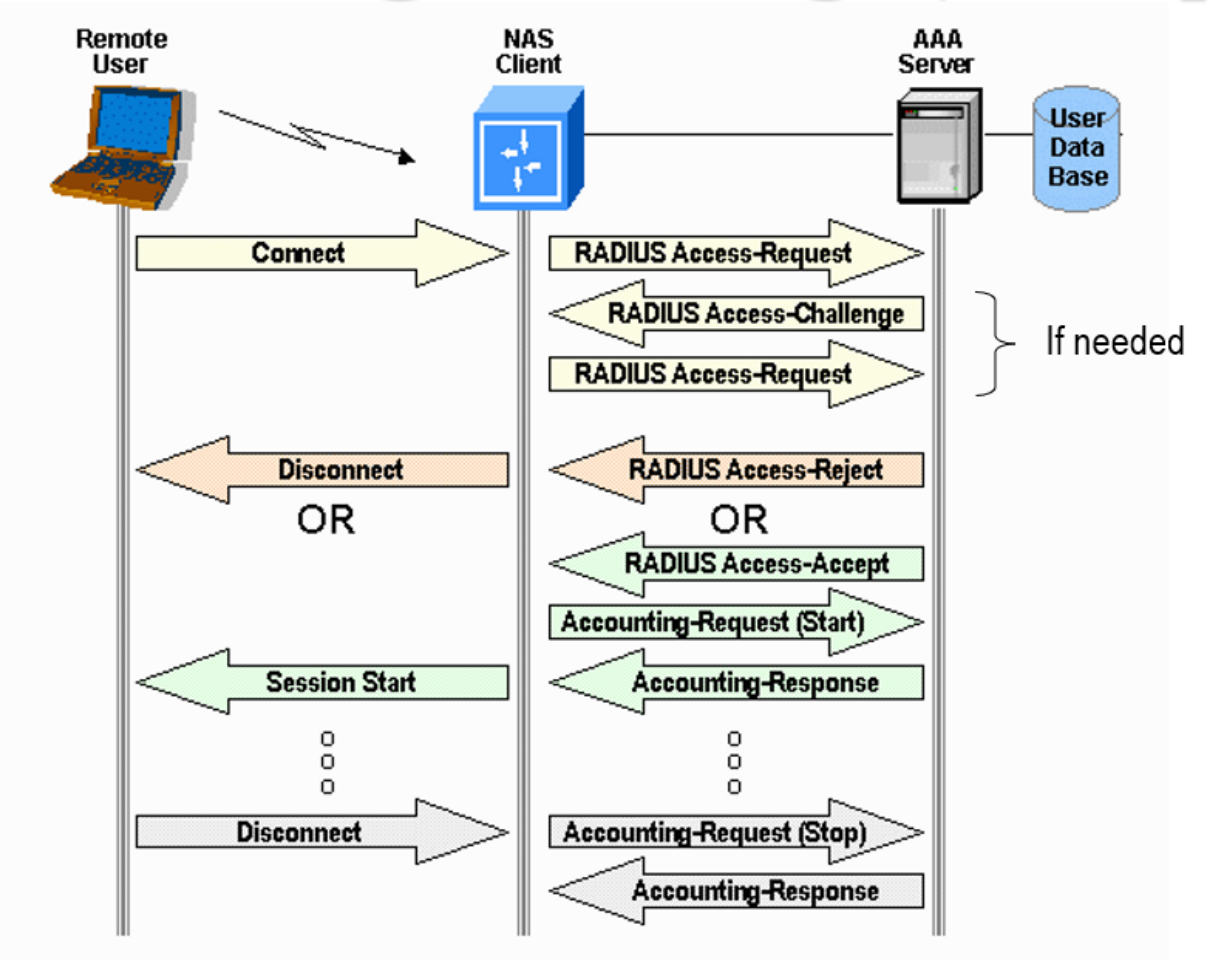
- **1812** per i messaggi di autenticazione e autorizzazione.
- **1813** per i messaggi di accounting (contabilizzazione).

Nota: Il server utilizza porte effimere per rispondere, come da standard architetturale C/S.

5.1 Scambio dei Messaggi (Message Exchange)

Il processo di autenticazione segue un diagramma a stati ben definito. Come mostrato nell'esempio del flusso di messaggi:

1. **Connect:** L'utente remoto si connette al NAS (livello fisico/datalink).
2. **Access-Request:** Il NAS invia la richiesta al Server contenente le credenziali dell'utente.
3. **Access-Challenge (Opzionale):** Se il metodo di autenticazione lo richiede (es. token OTP, smart card), il server non accetta/rifiuta subito ma invia una "sfida". Il NAS gira la richiesta all'utente e attende una nuova risposta.
4. **Decisione Finale:** Il server invia:
   - `Access-Reject`: Accesso negato, la porta sul NAS viene chiusa.
   - `Access-Accept`: Accesso consentito, vengono passati i parametri di configurazione.
5. **Accounting:** Solo *dopo* l'accettazione inizia la fase di contabilizzazione:
   - `Accounting-Request (Start)`: Segnala l'inizio della sessione.
   - Scambio dati utente...
   - `Accounting-Request (Stop)`: Segnala la fine della sessione e l'invio delle statistiche finali.

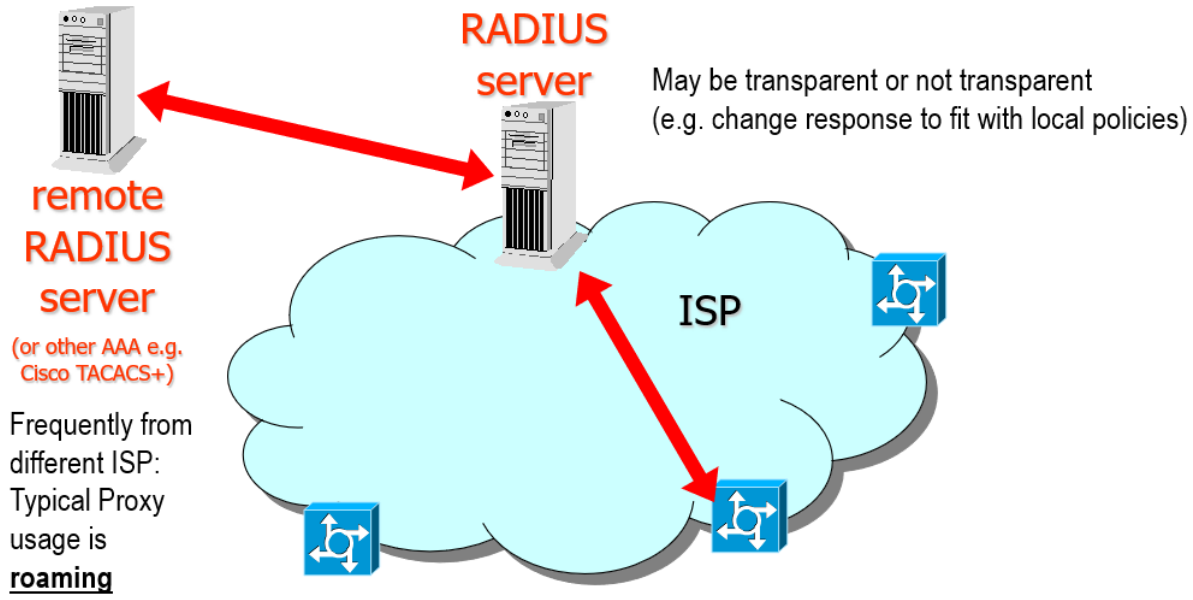


5.2 Architettura Proxy

Una delle caratteristiche più potenti di RADIUS è la capacità di operare tramite **Proxy**. In questo scenario, il server RADIUS locale (quello dell'ISP a cui l'utente è fisicamente connesso) non processa la richiesta, ma la inoltra a un **Remote RADIUS Server** (quello dell'organizzazione di appartenenza dell'utente).

Questo meccanismo è la base del **Roaming** (es. Eduroam) tra diversi ISP o organizzazioni:

- **Forwarding:** Il server locale agisce come un router di messaggi RADIUS.
- **Trasparenza:** Il proxy può essere trasparente (inoltra il messaggio così com'è) o non trasparente.
- **Policy Enforcement:** In modalità non trasparente, il proxy può modificare la risposta del server remoto per adattarla alle *local policies* (ad esempio, limitando la banda anche se il server remoto aveva autorizzato una velocità superiore).



6 Architettura del Server RADIUS

Il server RADIUS non è un semplice risponditore, ma un'applicazione complessa che orchestra diversi database per gestire il servizio AAA. L'architettura prevede la gestione separata di tre tipologie di dati:

Registered User Database: È il cuore dell'autenticazione. Per ogni entrata (identificata dallo *user_name*), il database deve contenere almeno:

- **Informazioni di Autenticazione:** I segreti (password, chiavi) necessari per verificare l'identità.
- **Metodo di Autenticazione:** Il protocollo specifico che l'utente deve usare (es. CHAP, PAP, EAP).
- **Attributi di Autorizzazione:** Il profilo di accesso specifico per quell'utente (es. quale indirizzo IP assegnare, quali filtri applicare).

Client Database: Questo è un punto critico spesso fonte di confusione. Questo database contiene l'elenco dei **NAS** (Network Access Server) autorizzati a interrogare il server RADIUS.

- **Attenzione:** Non bisogna confondere i "Client RADIUS" (i NAS) con gli "utenti finali". Se un NAS non è in questa lista (o non condivide il segreto corretto), le sue richieste verranno ignorate.

Accounting Database: Utilizzato quando RADIUS svolge funzioni di contabilizzazione. Spesso è separato dal database di autenticazione per motivi di performance e archiviazione, dato che raccoglie log di inizio/fine sessione e statistiche di traffico.

7 Sicurezza di RADIUS

La sicurezza del protocollo si basa su un meccanismo di fiducia tra NAS e Server, fondato su un **segreto condiviso** (Shared Secret) che non viene mai trasmesso sulla rete.

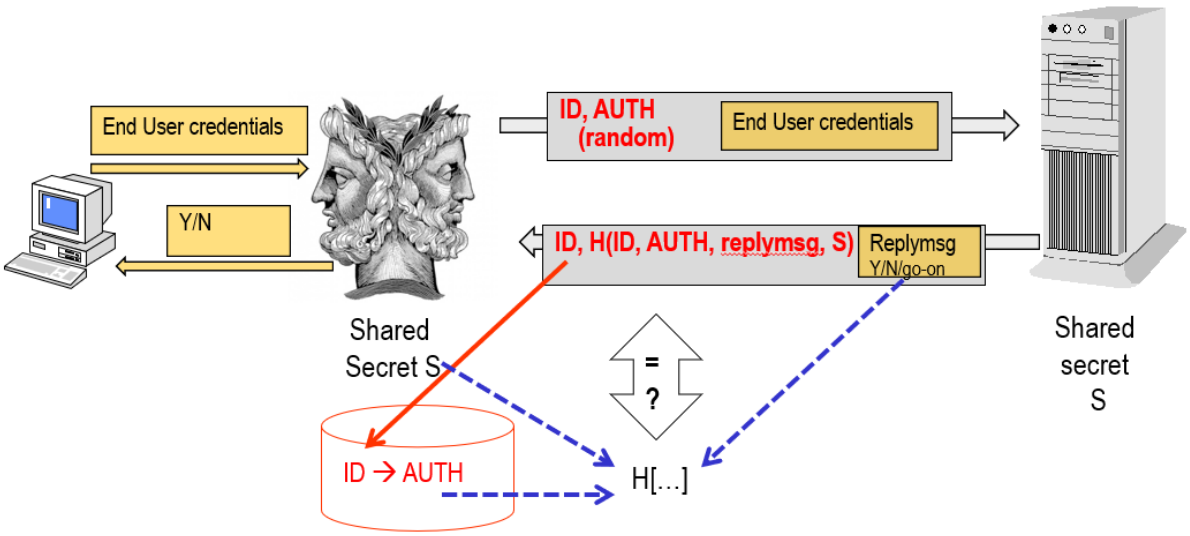
7.1 Meccanismo di Autenticazione della Risposta

RADIUS implementa un sistema di sicurezza asimmetrico nel flusso dei pacchetti:

- **Richiesta (Request):** Non è completamente autenticata.
- **Risposta (Reply):** È autenticata *per-packet*. Il NAS può verificare che la risposta provenga effettivamente dal server legittimo e non da un attaccante.

Come illustrato nello schema logico, il funzionamento è simile a una sfida (challenge-response):

1. Il NAS invia un *Request Authenticator* (un numero casuale) nella richiesta.
2. Il Server calcola un hash MD5 concatenando: l'ID del pacchetto, l'Authenticator originale, il messaggio di risposta e il **Shared Secret**.
3. Il NAS riceve la risposta e ricalcola l'hash. Se combacia, il server possiede il segreto ed è autentico.



7.2 Limitazioni e Criticità

Nonostante questo meccanismo, RADIUS presenta vulnerabilità storiche:

- **Algoritmo Debole:** Utilizza funzioni di hash **MD5** semplice, non HMAC (Hash-based Message Authentication Code), che è considerato crittograficamente più robusto.
- **Bassa Entropia:** Le chiavi condivise (Shared Keys) scelte dagli amministratori sono spesso parole semplici o brevi, rendendole vulnerabili ad attacchi a dizionario.
- **Crittografia Password:** La password dell'utente viene trasmessa cifrata, ma il meccanismo è "custom" e si basa anch'esso sulla stessa chiave condivisa MD5.

8 Formato del pacchetto RADIUS

I pacchetti RADIUS sono incapsulati in datagrammi UDP. La struttura dell'header è fissa e compatta:

Code (1 byte): Identifica il tipo di pacchetto. I valori decimali più comuni sono:

- **1:** Access-Request
- **2:** Access-Accept
- **3:** Access-Reject
- **4/5:** Accounting-Request / Response

Identifier (1 byte): Un contatore utilizzato per accoppiare le richieste con le relative risposte (fondamentale poiché UDP è connectionless).

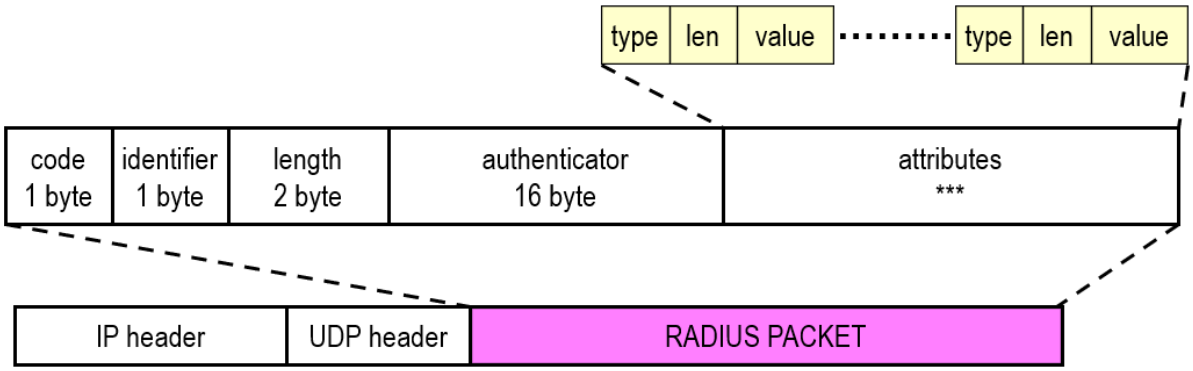
Length (2 byte): Indica la lunghezza totale del pacchetto. La dimensione varia da un minimo di 20 byte a un massimo di 4096 byte.

Authenticator (16 byte): Un campo cruciale a 128 bit.

- Nelle richieste funge da *nonce* (numero casuale) per cifrare la password.
- Nelle risposte viene usato per autenticare il server (come spiegato nella sezione sicurezza).

Attributes (Variabile): Questo campo segue la struttura **TLV** (Type-Length-Value) e trasporta i dati reali (User-Name, User-Password, IP-Address, etc.). *Nota critica:* Il campo "Type" è di solo 1 byte, limitando il numero di attributi possibili a 256. Questo si è rivelato un limite alla scalabilità ("not being extensible enough").

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

RADIUS non trasmette mai la password in chiaro, ma utilizza un meccanismo "custom" di offuscamento basato sull'algoritmo MD5 e sul segreto condiviso (Shared Secret) tra Client e Server.

Il processo, definito per il campo `User-Password`, avviene come segue:

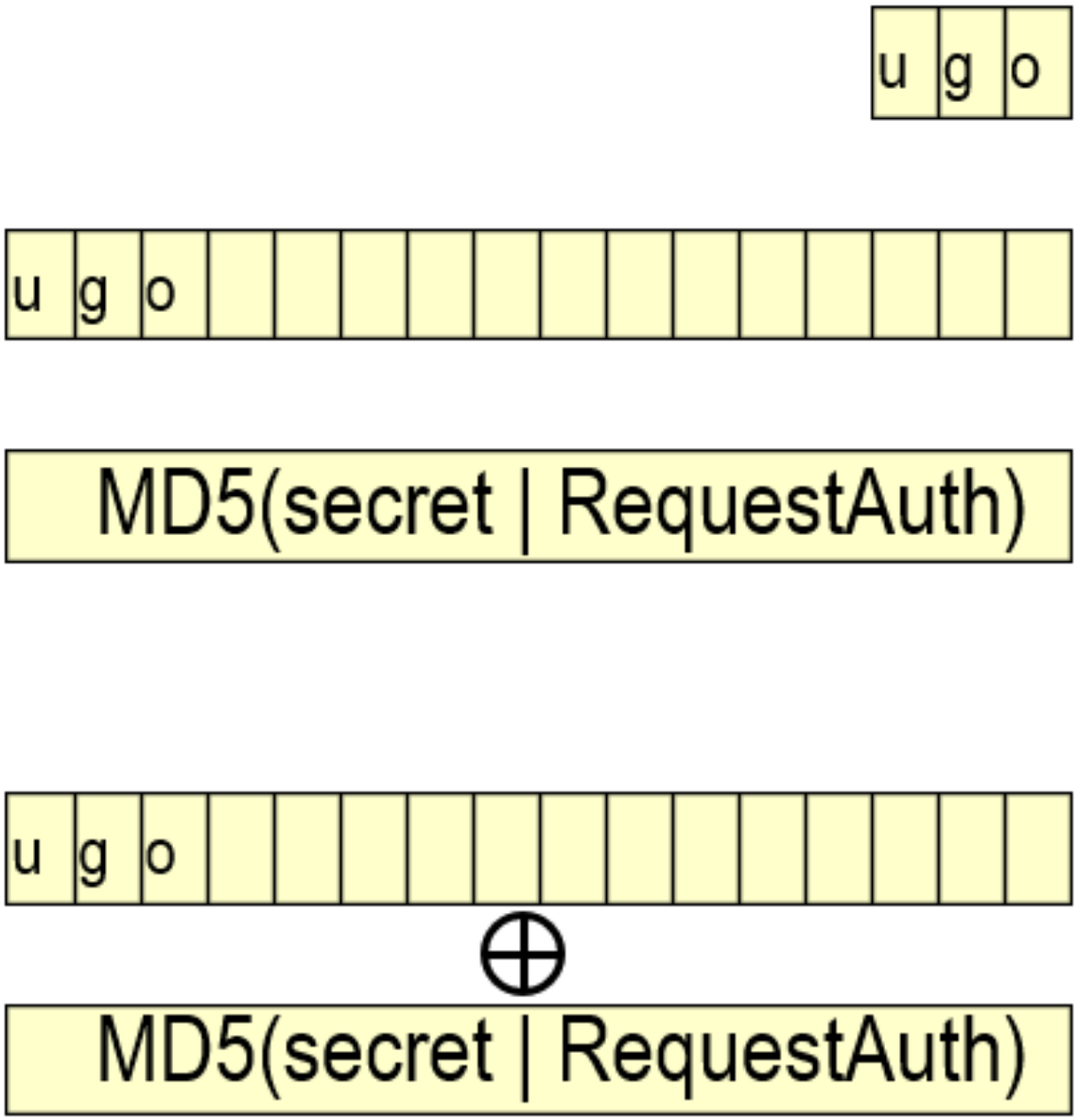
1. **Padding:** La password originale viene estesa (padded) con zeri finali affinché la sua lunghezza sia un multiplo esatto di 16 byte.

2. **Generazione Hash (Maschera):** Viene calcolato un hash MD5 concatenando il *Shared Secret* e il campo *Request Authenticator* (i 16 byte casuali presenti nell'header della richiesta).

$$H_1 = \text{MD5}(\text{Secret} + \text{RequestAuthenticator})$$

3. **XOR:** Viene effettuata un'operazione XOR bit a bit tra il primo blocco di 16 byte della password (paddata) e l'hash H_1 generato.

4. **Gestione Password Lunghe:** Se la password supera i 16 caratteri, il processo si ripete a catena. Il secondo blocco di password viene messo in XOR con l'MD5 calcolato sul *risultato cifrato del blocco precedente* (non più sul RequestAuthenticator).



10 Messaggi Principali

I pacchetti RADIUS trasportano attributi (TLV - Type Length Value) che definiscono i dettagli della richiesta e della risposta.

10.1 Access-Request

È il messaggio iniziale inviato dal NAS al Server. Contiene tipicamente:

User-Name: È obbligatorio e funge da chiave di ricerca primaria per accedere al database utenti.

User-Password: La password cifrata (o `CHAP-Password` se si usa quel protocollo).

NAS-Identifier / NAS-IP: Identifica quale NAS sta facendo la richiesta. Questo permette al server di limitare l'accesso di un utente solo a un sottoinsieme specifico di NAS.

NAS-Port: Indica la porta fisica (es. numero del modem in Dial-up) o logica (es. associazione Wi-Fi) da cui proviene l'utente. Utile per policy restrittive basate sulla porta.

10.2 Access-Accept

È la risposta positiva del server se le credenziali sono corrette. Non si limita a dire "OK", ma contiene tutti i parametri di **configurazione specifica del servizio**:

- **Service-Type:** Specifica il tipo di servizio autorizzato (es. *Framed* per PPP/SLIP, *Login* per accesso terminale).
- **Parametri di Rete:** Può includere l'indirizzo IP da assegnare all'utente, la maschera di sottorete, filtri di pacchetto, o l'assegnazione a una VLAN specifica.

10.3 Access-Reject

Indica che l'accesso è negato e la porta sul NAS deve essere bloccata. I motivi principali sono due:

1. **Fallimento Autenticazione:** Credenziali errate.
2. **Fallimento Autorizzazione:** Le credenziali sono corrette, ma la richiesta contiene attributi non accettabili (es. l'utente sta cercando di accedere da un NAS non permesso o fuori orario).

10.4 Access-Challenge

Questo messaggio introduce un'interazione più complessa. Viene usato quando il server necessita di ulteriori informazioni dall'utente prima di decidere.

Il flusso operativo è il seguente:

1. Il Server invia una `Access-Challenge` contenente un attributo `Reply-Message` (es. un testo di prompt o una sfida crittografica).
2. Il NAS mostra il messaggio all'utente e raccoglie la nuova risposta.
3. Il NAS invia una **NUOVA Access-Request** al server.

- Questa nuova richiesta ha un **Nuovo ID**.
- Il campo `User-Password` contiene la risposta dell'utente cifrata.

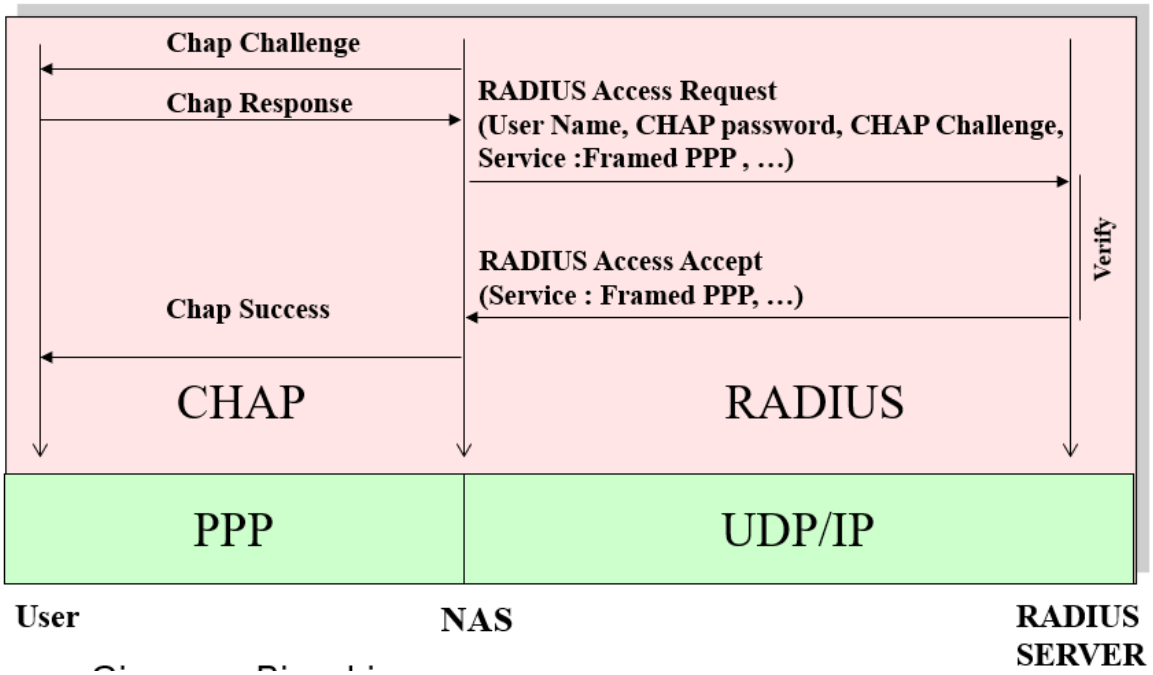
4. Il ciclo si ripete finché il server non invia Accept o Reject.

Questo meccanismo è utilizzato per token hardware, autenticazione GSM/UMTS o OTP.

10.5 Supporto PPP CHAP con RADIUS

RADIUS supporta nativamente il protocollo CHAP (*Challenge Handshake Authentication Protocol*), ma con una differenza architetturale importante:

- **Generazione della Challenge:** La "sfida" (Challenge) viene generata localmente dal **NAS**, non dal server RADIUS. Il NAS non ha bisogno di conoscere la password dell'utente per fare questo.
- **Verifica:**
  1. L'utente calcola la risposta e la invia al NAS.
  2. Il NAS incapsula la risposta CHAP e la Challenge originale in un pacchetto `Access-Request` verso il server RADIUS.
  3. Il Server RADIUS (che possiede la password in chiaro o reversibile nel DB) calcola la risposta attesa e la confronta con quella ricevuta.



11 Sicurezza e Vulnerabilità

Il protocollo RADIUS, essendo stato progettato molto tempo fa, presenta diverse debolezze intrinseche se non configurato correttamente o se non accompagnato da protocolli di sicurezza aggiuntivi (come IPSec o TLS).

11.1 Problemi principali

Le vulnerabilità strutturali includono:

- **Privacy limitata (Dati in chiaro):** Sebbene la password sia cifrata, molti attributi sensibili viaggiano in chiaro. Un attaccante che intercetta il traffico (sniffing) può leggere:
  - `User-Name`: Chi si sta connettendo.
  - `Calling-Station-ID`: Il numero di telefono o il MAC address dell'utente.
  - `NAS-IP-Address`: La posizione fisica o logica dell'utente nella rete.
- **Access-Request non autenticato:** Il pacchetto di richiesta inviato dal client non possiede una firma crittografica che ne validi l'integrità. Questo espone al rischio di:
  - *Forged Requests:* Un attaccante può generare pacchetti falsi sembrando un NAS legittimo.
  - *Modification / MITM:* Un attaccante attivo può modificare gli attributi della richiesta in transito senza che il server se ne accorga (a meno che non si usi Message-Authenticator).
- **Debolezza crittografica (MD5):** L'intero castello di sicurezza si basa sull'algoritmo di hashing MD5, oggi considerato obsoleto e vulnerabile alle collisioni, e su chiavi condivise (Shared Secrets) statiche.

11.2 Message-Authenticator

Per mitigare il problema delle richieste non autenticate, è stato introdotto un attributo di estensione (Type 80), che funge da firma digitale del pacchetto.

- **Funzione e Utilizzo:**
  - Serve a "firmare" l'intero pacchetto RADIUS, garantendo che non sia stato modificato durante il transito.
  - È fondamentale per i pacchetti `Access-Request` (che nativamente non hanno protezione di integrità), ma può essere incluso in qualsiasi pacchetto.

– È **OBBLIGATORIO** quando si utilizza il protocollo **EAP** (Extensible Authentication Protocol) su RADIUS (RFC 2869), per proteggere lo scambio di chiavi sensibili.

- **Struttura e Calcolo:** L'attributo contiene un valore HMAC-MD5 di 16 byte.

Type=80	Len=18	Authenticator (16 byte)

Il calcolo differisce dalla classica firma RADIUS perché utilizza **HMAC** (Hash-based Message Authentication Code), che è più robusto del semplice hash concatenato:

$$\text{Authenticator} = \text{HMAC-MD5}(\text{Intero Pacchetto}, \text{Shared Secret})$$

Nota tecnica: Durante il calcolo dell'hash, il campo "Authenticator" all'interno del pacchetto viene temporaneamente riempito con zeri ($00 \ldots 00$).

11.3 Attacchi noti

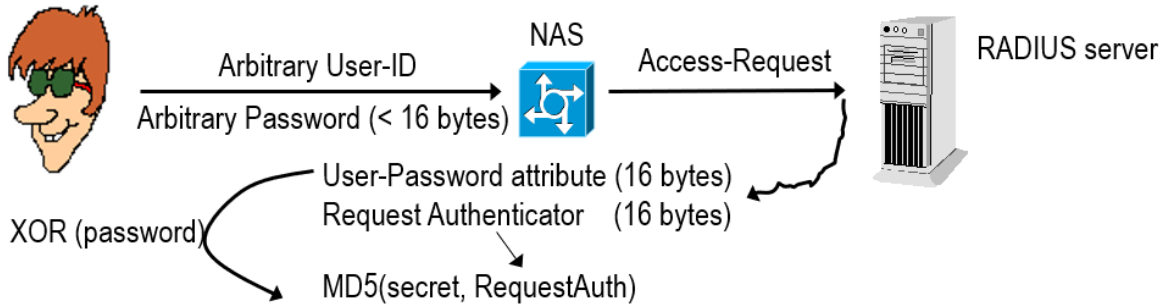
La sicurezza di RADIUS crolla se il *Shared Secret* è debole.

- **Attacco a Dizionario (Dictionary Attack):** Questo attacco offline mira a recuperare il segreto condiviso.
  - **Causa:** Spesso gli amministratori usano segreti ASCII semplici (parole di dizionario) e, errore ancora più grave, utilizzano lo stesso segreto per tutti i NAS della rete.
  - **Meccanismo:** L'attaccante cattura una coppia `Access-Request` / `Access-Response`.
  - **La Falla:** L'attaccante conosce tutti i dati del pacchetto tranne il segreto. La formula di verifica è:

$$\text{RespAuth} = \text{MD5}(\text{Code} + \text{ID} + \text{Len} + \text{ReqAuth} + \text{Attr} + \textbf{Secret})$$

Poiché il segreto è posizionato alla *fine* della stringa, l'attaccante può precalcolare lo stato dell'MD5 per la parte fissa del pacchetto, rendendo l'attacco di forza bruta estremamente veloce.

  - **Variante:** È possibile condurre l'attacco anche possedendo solo una Request che contiene una password utente nota.

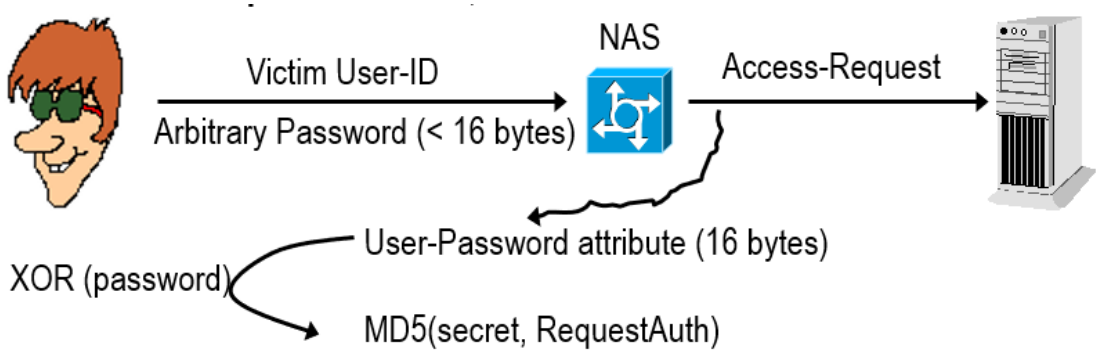


- **Password Cracking (Attacco Attivo):** Anche senza conoscere il segreto condiviso, un attaccante può tentare di indovinare la password di un utente legittimo sfruttando la debolezza del protocollo di cifratura XOR e l'assenza di controlli lato server.

  FASE 1: Furto del Keystream (Mask) L'attaccante invia una richiesta al server (fingendosi un NAS o tramite un NAS compromesso) per l'utente vittima, ma inserendo una password arbitraria nota (es. "A").

  – L'attaccante calcola il campo `User-Password` inviato: $C = P_{arbitraria} \oplus K$, dove $K = \text{MD5}(\text{Secret}, \text{RequestAuth})$.

  – Conoscendo C e $P_{arbitraria}$, l'attaccante ricava K (il keystream o "maschera") facendo: $K = C \oplus P_{arbitraria}$.

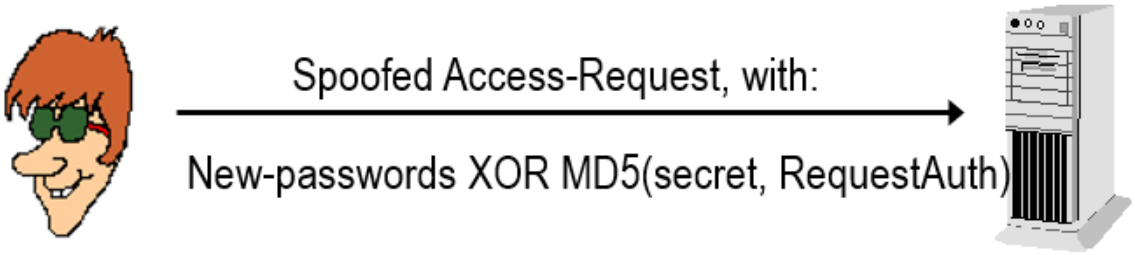


FASE 2: Brute Force Online Ora l'attaccante possiede la maschera crittografica K associata a quel specifico `RequestAuthenticator`.

1. L'attaccante può ora cifrare qualsiasi password candidata (P') usando la maschera rubata: $C' = P' \oplus K$.
2. Invia nuove richieste al server (Spoofed Access-Request) riutilizzando lo **stesso** RequestAuthenticator della Fase 1.

3. **Vulnerabilità sfruttata:** I server RADIUS tipicamente *non controllano* se un RequestAuthenticator è già stato usato (Replay). Inoltre, spesso non c'è limite al numero di tentativi di autenticazione lato server.
4. L'attaccante prova password diverse finché il server non risponde `Access-Accept`.

Nota: Questo attacco funziona efficacemente per password inferiori ai 16 byte.



12 PRNG (Pseudo-Random Number Generators)

La sicurezza dell'intero protocollo RADIUS dipende fortemente dall'unicità e dall'imprevedibilità del campo **Request Authenticator**. Se questo numero non è realmente casuale, l'intero schema di sicurezza crolla.

12.1 Implementazioni

Molte implementazioni storiche o di bassa qualità soffrono di gravi difetti nella generazione di questi numeri:

- **Requisito di Unicità:** Il Request Authenticator deve fungere da *nonce* (Number used ONCE). Non deve mai ripetersi per la stessa chiave condivisa.
- **Problema dei PRNG Deboli:**
  - Spesso vengono utilizzati generatori non crittografici (come la funzione `rand()` standard delle librerie C), che sono progettati per simulazioni statistiche, non per la sicurezza.
  - Questi generatori hanno cicli brevi (si ripetono presto) e sono prevedibili.
- **Le tre verifiche fondamentali:** Quando si analizza un PRNG in un'implementazione RADIUS, bisogna porsi tre domande:
  - **Ciclicità:** Quanto è lungo il ciclo prima che la sequenza si ripeta?
  - **Prevedibilità:** Se un attaccante conosce un valore estratto, può calcolare matematicamente i successivi? (Esempio: nei generatori congruenziali lineari - LCG, basta un valore per conoscere tutta la sequenza futura).

- **Unicità vs Ripetizione:** Alcuni generatori come il *Mersenne Twister* hanno un ciclo enorme ($2^{19937} - 1$), ma i valori all'interno del ciclo possono ripetersi, creando collisioni fatali per RADIUS.

12.2 Note matematiche e Attacco del Compleanno

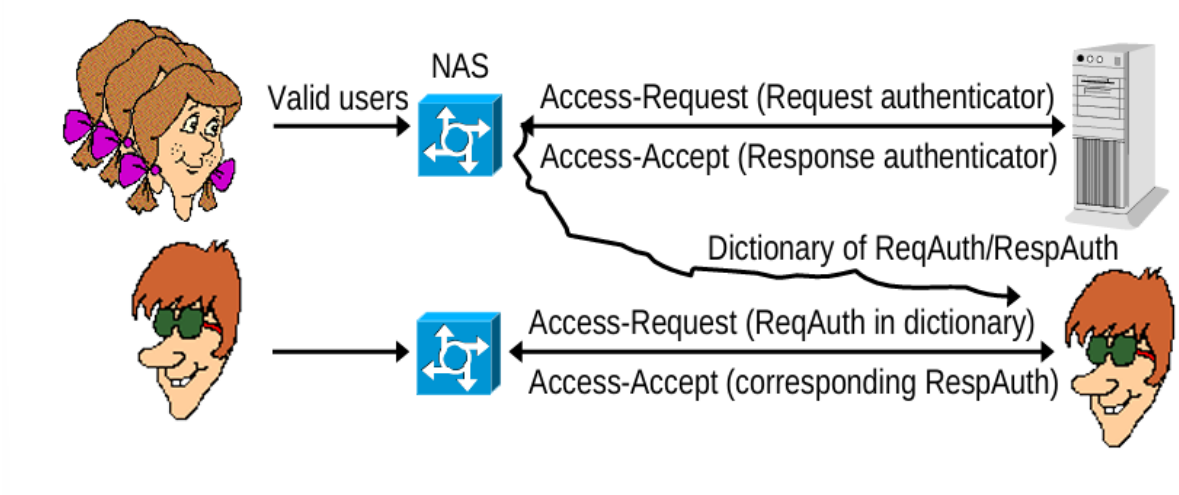
La probabilità che un Request Authenticator si ripeta (collisione) segue il paradosso del compleanno, rendendo i cicli apparentemente lunghi molto più vulnerabili di quanto sembri.

- **Il Paradosso del Compleanno:** Assumendo un ciclo del PRNG pari a 2^N, non serve attendere 2^N pacchetti per trovare una collisione. La probabilità del 50% di trovare un duplicato si raggiunge con circa $\sqrt{2^N} = 2^{N/2}$ pacchetti.
- **Esempi pratici di vulnerabilità:**
  - Se $N = 16$ bit: bastano appena **250 pacchetti** per avere un'alta probabilità di collisione!
  - Se si usa `drand48` (48 bit): l'attacco riesce con circa 16 Milioni di pacchetti. In una rete ad alto traffico, questo avviene in tempi molto brevi.
- **Esempio di "Cattiva Implementazione":** Una tecnica errata per tentare di aumentare l'entropia consiste nel concatenare più numeri casuali deboli.
  - Si parte da un *seed* o numero casuale a 32 bit.
  - Si genera l'autenticatore a 128 bit concatenando 4 chiamate consecutive al generatore.
  - **Risultato:** Invece di aumentare la sicurezza, il ciclo effettivo si riduce drasticamente (es. a 30 bit), rendendo possibile un attacco con soli **32.000 pacchetti**.

12.3 Replay attacks

Se il Request Authenticator è prevedibile o viene riutilizzato (a causa dei problemi PRNG sopra descritti), è possibile effettuare attacchi di replay per autenticare o autorizzare utenti illegalmente.

Il concetto base è che la risposta del server (Accept/Reject) è firmata basandosi sul Request Authenticator della richiesta. Se un attaccante può prevedere o riutilizzare un Authenticator per il quale possiede già una risposta catturata in precedenza, può ingannare il client o il server.

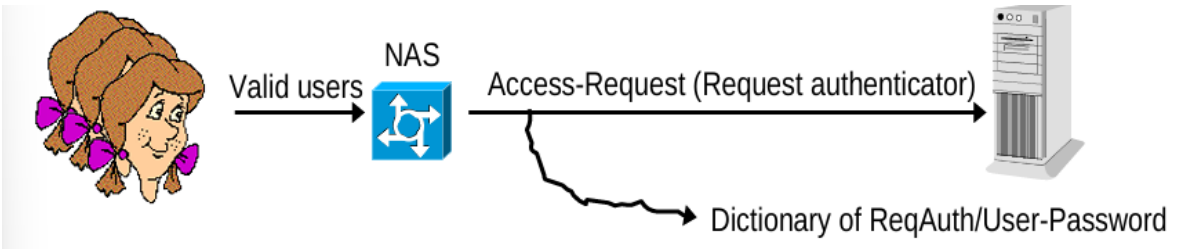


13 Attacchi alle Password (Decrittazione)

La vulnerabilità più critica legata alla ripetizione dei Request Authenticator riguarda la decifratura delle password degli utenti. Poiché la cifratura della password in RADIUS è un semplice XOR con un keystream generato da MD5(Secret, RequestAuth), se il RequestAuth si ripete, il keystream è identico.

13.1 Attacco Passivo (Monitoraggio)

Un attaccante passivo che monitora la rete (sniffer) costruisce un dizionario di coppie $\{\text{RequestAuthenticator}, \text{EncryptedPassword}\}$.



- **Rilevamento Collisione:** L'attaccante attende di vedere un Request Authenticator già presente nel suo dizionario.
- **Analisi XOR:** A questo punto, l'attaccante possiede due password cifrate (C_1 e C_2) con la stessa chiave (K).

$$C_1 = P_1 \oplus K$$

$$C_2 = P_2 \oplus K$$

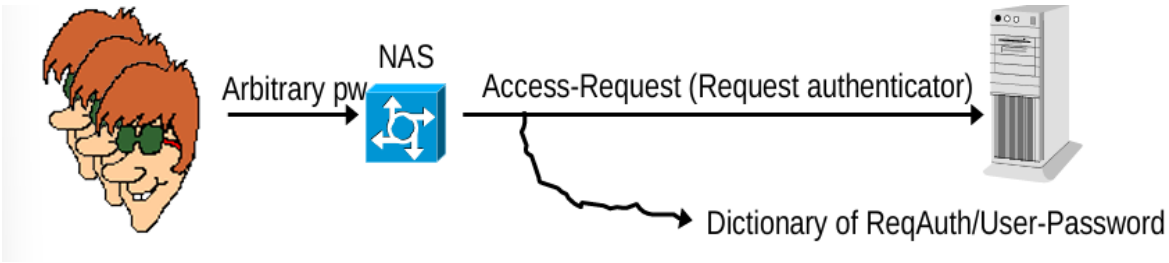
Facendo lo XOR tra i due testi cifrati, la chiave si elide:

$$C_1 \oplus C_2 = (P_1 \oplus K) \oplus (P_2 \oplus K) = P_1 \oplus P_2$$

- **Estrazione Dati:**
  - Si ottiene lo XOR delle due password in chiaro.
  - Poiché le password hanno lunghezze diverse, la parte finale della password più lunga viene messa in XOR con il padding (zero) della più corta, **rivelando i caratteri in chiaro**.
  - Conoscendo la lunghezza e parziali informazioni, è banale ricostruire entrambe le password (specialmente se a bassa entropia).

13.2 Attacco Attivo (Chosen Plaintext)

L'attacco passivo richiede tempo per attendere una collisione naturale. L'attacco attivo accelera il processo inserendo deliberatamente voci nel "dizionario" dell'attaccante.



- **Procedura:** L'attaccante invia continuamente al server RADIUS delle `Access-Request` false.
- **Payload Noto:** In queste richieste, l'attaccante usa una password scelta e nota (es. "password123") e un Request Authenticator generato (che spera collliderà in futuro o che ha predetto).
- **Costruzione del Dizionario:** Anche se il server risponde `Access-Reject`, la richiesta è stata inviata. L'attaccante ora conosce:
  - Il `Request Authenticator` usato.
  - La password in chiaro P_{known}.
  - La password cifrata inviata C.

  Da questo può calcolare il Keystream $K = P_{known} \oplus C$.
- **Decifratura Immediata:** Se un utente legittimo invia una richiesta con quello stesso Authenticator (collisione), l'attaccante possiede già la chiave K per decifrare istantaneamente la password dell'utente.
- **Vantaggio:** Non serve indovinare la password dell'utente; si usa il meccanismo di autenticazione stesso per "pre-calcolare" le chiavi di cifratura basandosi sui tentativi falliti.

14 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.

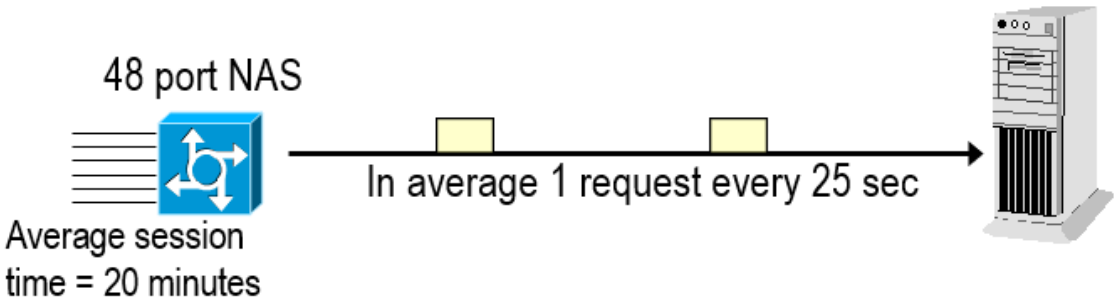
15 Radius: Oggi

Nonostante le sue origini datate, RADIUS ha subito un'evoluzione notevole nel suo utilizzo pratico:

- **Origini e Stato Attuale:** Inizialmente, il protocollo fu sviluppato con uno scopo limitato: supportare l'accesso di utenti *dial-up* (modem analogici) tramite protocollo PPP e terminali di login testuali. Tuttavia, col tempo, si è affermato come lo **standard de-facto** per tutti i servizi AAA, godendo di un supporto universale da parte di tutti i venditori di apparati di rete (Cisco, Juniper, Huawei, etc.).
- **Limiti Funzionali Critici:** Sebbene sia onnipresente, l'architettura originale di RADIUS mostra oggi gravi segni di invecchiamento rispetto alle esigenze moderne:
  - **Scalabilità:** Il protocollo era pensato per i primi ISP con poche migliaia di utenti. Oggi deve gestire le reti di operatori mobili nazionali con **milioni di utenti** simultanei, mettendo sotto stress l'architettura centralizzata.
  - **Estensibilità:** L'introduzione di nuove tecnologie (Wireless, DSL, 4G/5G, Ethernet) e scenari di servizio complessi (come il Mobile-IP, il Roaming internazionale o il billing "carrier-grade") ha forzato RADIUS oltre i suoi limiti originali. Spesso mancano i campi (attributi) standard per gestire queste nuove funzionalità.
  - **Interoperabilità:** Molte funzioni vitali per una rete moderna (come il failover tra server, il bilanciamento del carico o la selezione intelligente del server) non sono state specificate nello standard originale. Di conseguenza, ogni produttore ha implementato soluzioni proprietarie, rendendo difficile far comunicare apparati di marche diverse.

15.1 Scalabilità: Un Esempio Pratico

Per comprendere concretamente il problema della scalabilità, analizziamo uno scenario tipico di un ISP moderno.



- **Scenario Base:** Immaginiamo un singolo NAS a 48 porte. Con una durata media della sessione utente di 20 minuti, questo genera un carico irrisorio: circa 1 richiesta ogni 25 secondi verso il server.
- **Il Problema dei Grandi Numeri:** Se scaliamo questo scenario a una rete reale con **10.000 NAS**:
  - Il server viene investito da **400 Access-Request al secondo**.
  - Se ogni pacchetto è di circa 1KB, questo genera un traffico costante di 3.2 Mbps.
  - A questo bisogna sommare altre 400 richieste di *Accounting* al secondo, raddoppiando di fatto il carico transazionale.
- **Rischi di Congestione e Packet Loss:** Molte implementazioni server (e l'infrastruttura di rete stessa) faticano a gestire questo volume di richieste UDP senza perdere pacchetti. Poiché UDP non gestisce la congestione, il server rischia di collassare sotto il carico.
- **L'effetto Valanga (Load Peaks):** La situazione diventa critica in caso di riavvio simultaneo dei NAS (ad esempio dopo un blackout in un quartiere). Al ripristino della corrente:
  - Migliaia di record di accounting (Stop/Start) vengono inviati back-to-back.
  - Tutti gli utenti tentano il login simultaneamente.

Conclusione: In questi scenari, i server AAA e le reti che li ospitano vanno in saturazione (Congestion), causando la perdita di pacchetti e il disservizio per l'utente finale.

16 Standardizzazione IETF

Per affrontare questi limiti, l'IETF (Internet Engineering Task Force) ha avviato due percorsi paralleli di standardizzazione:

- **Diameter (L'approccio Rivoluzionario):**
  - Nato nel 1998, è stato progettato come successore diretto di RADIUS per superarne i limiti strutturali.
  - Attualmente gestito dal gruppo di lavoro DiME (Diameter Maintenance and Extensions).
  - È stato costruito nativamente per supportare le reti moderne: gestisce Mobile IPv6, Quality of Service (QoS), e applicazioni complesse come la telefonia IP (SIP) e la gestione delle VLAN.
- **RADext (L'approccio Evolutivo):**
  - Nato nel 2003, questo gruppo di lavoro si focalizza sull'estendere RADIUS mantenendo però la **retro-compatibilità obbligatoria**.
  - L'obiettivo è permettere a RADIUS di supportare scenari moderni (VoIP, LAN/VLAN, sistemi pre-pagati) senza dover sostituire l'infrastruttura esistente.
  - *Nota importante:* RADext non modifica il livello di trasporto (che rimane UDP) né risolve le debolezze di sicurezza intrinseche; si limita ad aggiungere nuovi attributi e funzionalità applicative.
- **Compatibilità RADIUS/Diameter:**
  - Entrambi i gruppi di lavoro hanno come obiettivo fondamentale garantire meccanismi di traduzione e coesistenza, permettendo una migrazione graduale da RADIUS a Diameter.

17 Perché "duplicare" il lavoro?

Nel panorama dei protocolli di rete, sorge spontanea una domanda: perché investire risorse nello sviluppo di un nuovo protocollo (Diameter) quando ne esiste già uno universalmente accettato (RADIUS)? La risposta risiede nella necessità di gestire scenari moderni senza abbandonare l'eredità del passato.

- **Il Ruolo di RADIUS (Lo standard "Legacy"):** RADIUS non può essere semplicemente dismesso.
  - È lo standard *de-facto* per l'accesso aziendale e ISP tradizionale. Esiste un "lavoro massivo" già svolto in termini di deployment hardware e software.
  - È profondamente integrato nei domini di business: i database utenti, i sistemi di fatturazione e gli apparati di rete sono costruiti attorno ad esso.

  - Le sue limitazioni sono state storicamente aggirate tramite estensioni "ad-hoc" proprietarie, che però hanno creato frammentazione.

- **Il Ruolo di Diameter (L'evoluzione necessaria):** Diameter non è solo un aggiornamento, ma un protocollo completamente nuovo progettato per il futuro.
  - Sebbene mantenga una logica di compatibilità all'indietro (i concetti base rimangono), cambia radicalmente le fondamenta: il trasporto, il formato dei pacchetti e la gestione degli errori.
  - È decisamente più potente e scalabile, ma il prezzo da pagare è una complessità molto superiore (*Diameter = 2 x RADIUS*).
  - È la scelta obbligata per gli scenari emergenti, in particolare per le reti mobili 3G/4G/5G (LTE, IMS), dove la mobilità e la qualità del servizio sono critiche.

- **Conclusione Strategica:** Non ci sarà un vincitore unico. L'incertezza sul futuro porta a una strategia di coesistenza: si mantiene RADIUS per le reti LAN/Enterprise e si adotta Diameter per il Core Network degli operatori mobili.

18 Evoluzione: Diameter

Diameter nasce specificamente per risolvere le carenze strutturali di RADIUS che ne impedivano l'uso affidabile su larga scala.

- **Trasporto Affidabile (TCP/SCTP vs UDP):** Mentre RADIUS si affida a UDP (che non garantisce la consegna), Diameter richiede un trasporto affidabile.
  - Questo è cruciale per l'Accounting: in un operatore telefonico, la perdita di un pacchetto di accounting equivale a una perdita diretta di denaro (*Packet loss = money loss*).
  - Il protocollo preferito è **SCTP** (Stream Control Transmission Protocol), che permette di gestire una connessione persistente tra client e server, supportando il *pipelining* dei messaggi (inviare più richieste senza attendere la risposta della precedente) senza i problemi di blocco tipici di TCP.

- **Gestione Standardizzata dei Guasti (Failover):** In RADIUS, il comportamento di un NAS quando un server non rispondeva era lasciato all'implementazione del produttore. Diameter standardizza questo processo.
  - Include un meccanismo di **Watchdog** a livello applicativo: i peer si scambiano messaggi periodici per confermare di essere "vivi".

- Il timer di default è di 30 secondi (±2 secondi di *jitter* casuale per evitare che tutti i dispositivi inviino il watchdog nello stesso istante, sincronizzandosi). Se arrivano pacchetti dati normali, il timer si resetta, ottimizzando il traffico.

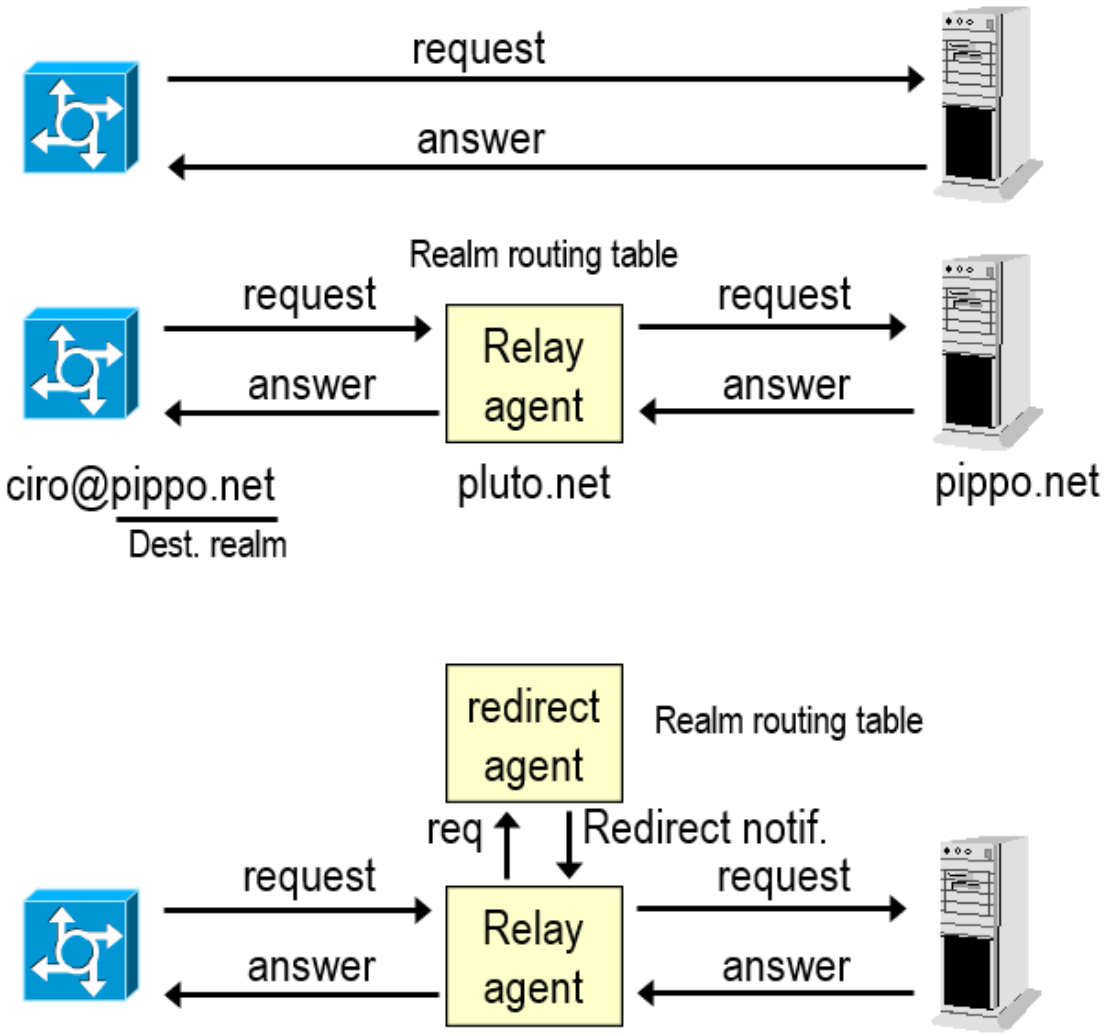
18.1 Estensione dei limiti funzionali

Diameter rimuove i "colli di bottiglia" che limitavano la scalabilità di RADIUS:

- **Spazio degli Indirizzi e Attributi:**
  - *AVP (Attribute-Value-Pair):* Il campo che definisce il tipo di attributo passa da 8 bit (max 255 tipi, ormai esauriti in RADIUS) a **32 bit** in Diameter, garantendo estensibilità infinita.
  - *Identificatori di Pacchetto:* RADIUS usa un ID a 8 bit per accoppiare richieste e risposte, limitando a 256 le richieste "in volo" (pending) per server. Diameter introduce un identificatore **Hop-by-Hop a 32 bit**, permettendo milioni di richieste simultanee.
- **Rilevamento Duplicati:** Grazie all'identificatore *End-to-End* e al flag 'T' (Retransmission), Diameter permette al server di capire univocamente se un pacchetto è un duplicato, vitale per evitare doppi addebiti nell'accounting.
- **Negoziazione delle Capacità (Capability Negotiation):** RADIUS tende a rifiutare silenziosamente o genericamente richieste che contengono attributi non capiti. Diameter introduce un sistema più intelligente:
  - Gli attributi hanno un flag **'M'** (Mandatory).
  - Se il server non comprende un attributo opzionale, può ignorarlo e procedere. Rifiuta la richiesta solo se manca la capacità di gestire un attributo marcato come "Mandatory".

19 Funzionamento degli Agenti Diameter

A differenza dell'architettura client-server pura di RADIUS, Diameter definisce in modo formale il ruolo degli intermediari nella rete, introducendo concetti fondamentali per il roaming.



Esistono tre tipologie principali di agenti intermedi:

Relay Agent (Il "Postino"): Si limita a instradare i messaggi. Legge le informazioni di routing (come il *Destination-Realm*, es. `ciro@pippo.net`) e inoltra la richiesta al server competente. **Nota:** Non ispeziona né modifica il contenuto del messaggio (payload), mantenendo l'integrità originale.

Proxy Agent (L'Intermediario Attivo): Simile al Relay, ma con poteri di modifica. Può aggiungere, rimuovere o cambiare i valori degli attributi (AVP) per applicare

policy locali (es. forzare un determinato QoS). **Implicazione di sicurezza:** Poiché modifica il messaggio, rompe l'autenticazione End-to-End e la firma digitale originale del client.

Redirect Agent (Il "Segnalatore"): Non inoltra il messaggio. Quando riceve una richiesta, risponde al mittente con un'informazione di reindirizzamento, dicendo sostanzialmente: *"Non chiedere a me, il server corretto per questa richiesta si trova a questo indirizzo"*. È utile per centralizzare le directory di routing senza dover gestire tutto il traffico dati attraverso un unico punto.