Server RADIUS: Architettura e Sicurezza

Il protocollo AAA per il controllo degli accessi

Prof. Capolupo

3 marzo 2026

Introduzione al protocollo RADIUS

RADIUS (Remote Authentication Dial-In User Service) è un protocollo di rete che fornisce una gestione centralizzata di **Authentication, Authorization, e Accounting (AAA)**.

- **Standard RFC:** Definito originariamente in RFC 2865 (Auth) e RFC 2866 (Accounting).
- **Livello Trasporto:** Utilizza **UDP** (Porte 1812 per Auth e 1813 per Accounting).
- **Modello Client/Server:** Il NAS (Network Access Server) agisce come client verso il server RADIUS.

Il Cuore del Sistema: Il Modello AAA

Basandosi sulle specifiche Cisco, RADIUS implementa tre funzioni distinte:

1. **Authentication:** Verifica l'identità dell'utente (Username/Password, Challenge/Response).
2. **Authorization:** Determina quali servizi l'utente è autorizzato a utilizzare (es. accesso a VLAN specifiche).
3. **Accounting:** Traccia il consumo di risorse (tempo di connessione, volume dati) per audit o fatturazione.

Sicurezza e Shared Secret

La sicurezza tra il Client (es. un Router o uno Switch Cisco) e il Server RADIUS si basa su:

- **Shared Secret:** Una stringa alfanumerica configurata su entrambi i nodi. Non viene mai inviata sulla rete.
- **Cifratura delle Password:** Solo l'attributo "User-Password" viene cifrato tramite un hash MD5 che combina la password e lo Shared Secret.
- **Integrità dei Pacchetti:** Ogni risposta del server contiene un "Response Authenticator" per prevenire attacchi di tipo replay.

Integrazione Moderna: Da RADIUS a REST

Perché usare JS/REST con RADIUS?

- **Flessibilità:** Permette a un'applicazione Web o Mobile di autenticare gli utenti contro un database aziendale centralizzato (Active Directory/FreeRADIUS).
- **Interoperabilità:** Trasforma un protocollo legacy UDP in una moderna API JSON.
- **Caso d'uso:** Portale captive per WiFi aziendale o autenticazione VPN tramite Dashboard Web.

Esempio Pratico: Autenticatore REST in Node.js

Sfruttando la libreria `node-radius`, possiamo creare un endpoint che valida le credenziali.

```
const express = require('express');
const radius = require('node-radius');
const app = express();
app.use(express.json());

const RADIUS_CONFIG = {
  host: '192.168.1.100', // IP del Server RADIUS
  secret: 'segreto_condiviso_cisco',
  port: 1812
};

app.post('/api/auth', (req, res) => {
  const { username, password } = req.body;

  // Creazione pacchetto Access-Request
  const client = radius.createClient(RADIUS_CONFIG);
  client.accessRequest({
    attributes: [
      ['User-Name', username],
      ['User-Password', password]
    ]
  }, (err, response) => {
    if (response.code === 'Access-Accept') {
      res.status(200).json({ status: "success", message: "Autenticato!" });
    } else {
      res.status(401).json({ status: "fail", message: "Negato" });
    }
  });
});
```

Conclusioni

- RADIUS rimane lo standard di fatto per l'autenticazione di rete scalabile.
- L'integrazione con **Node.js e REST** permette di modernizzare l'accesso alle risorse legacy, rendendo il controllo degli accessi fruibile da qualsiasi applicazione moderna.