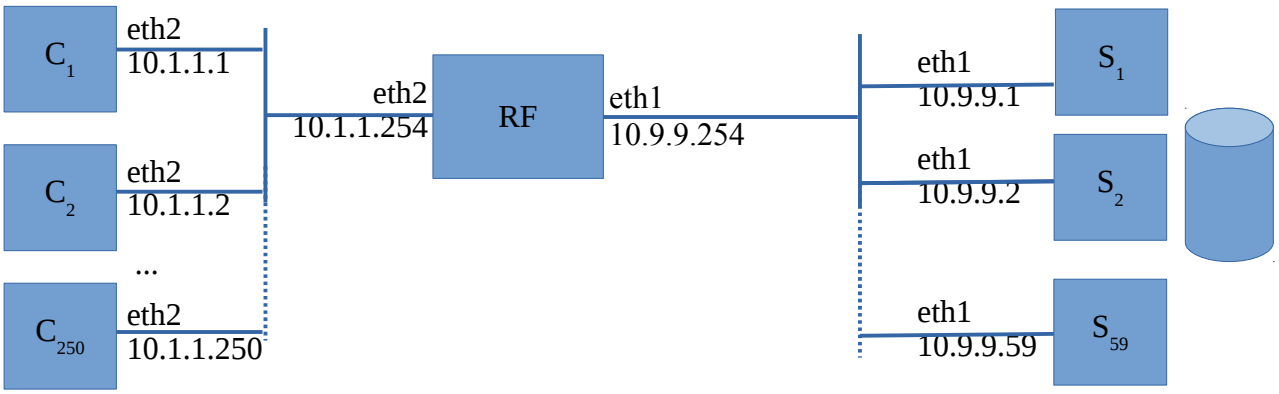
Laboratorio di Amministrazione di Sistemi T

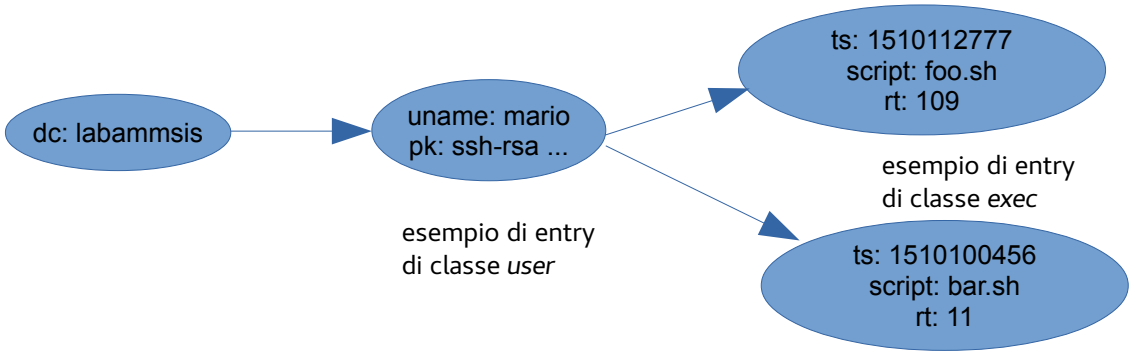
Prova pratica del 13 giugno 2018

Descrizione generale del problema



Il sistema di "scripting as a service" rappresentato in figura prevede un insieme di 250 Client, 59 Server, e un Router-Firewall (RF) tra le rispettive reti. Il RF media le richieste di esecuzione remota di codice inviate dai client ai server.

La base di utenti autorizzata all'uso degli script sui server è registrata sulla directory LDAP ospitata da RF. Le entry servono sia a rappresentare le identità degli utenti che a tenere traccia delle richieste da essi effettuate, come nell'esempio di struttura qui riportato:



Il RF utilizza la base dati LDAP per garantire che gli account siano allineati su tutti i client e i server.
I server montano le home degli utenti via NFS come /home/*uname*
L'utente *root* di RF ha accesso via ssh a tutte le macchine client e server.

File da consegnare
(tra parentesi quadre la collocazione e il punteggio, non fanno parte del nome)

activity.schema.ldif [RF, 4] – Definire i tipi di attributo *uname*, *pk* e *script* (stringhe), *ts* e *rt* (interi), e le classi *user* e *exec* che li utilizzino come in figura. Le entry di classe *user* hanno i due attributi *uname* che rappresenta uno username e *pk* che contiene la sua chiave pubblica; le entry di classe *exec* hanno tre attributi: *ts* rappresenta il timestamp univoco di inizio e *rt* la durata di un'esecuzione di *script* (da parte dell'*uname* da cui dipende la entry di classe *exec*)

adduser.sh [client, 9] – Questo script viene usato sui client per creare utenti. Lo username viene passato come parametro; lo script:

- invoca il comando di sistema per creare l'utente Unix
- genera per esso una coppia di chiavi SSH di tipo RSA
- invia attraverso *rsyslog* username e chiave pubblica a RF.

Indicare nei commenti come configurare rsyslog sui client e su RF perché tali messaggi inviati dai client vengano scritti nel file `/var/log/newusers` di RF

id-sync.sh [RF, 13] – Questo script osserva senza sosta il file `/var/log/newusers` e per ogni nuovo messaggio

- inserisce una nuova entry di classe *user* in LDAP coi dati corrispondenti
- avvia un ciclo che su ognuno dei 59 server, via ssh
  - crea l'utente
  - ne configura l'account perché sia accessibile dal client con autenticazione RSA
  - configura la sua tabella di cron per eseguire *script.sh* dalla home dell'utente, ogni ora al minuto corrispondente all'indice del server (es. per S_{12} al minuto 12)

(si trascuri la necessità di clonazione dell'utente sui client – gli utenti sui client utilizzeranno questa configurazione per depositare script nella directory `~/jobs/` *del filesystem dei server)*

fwinit.sh [RF, 10] – Questo script configura il packet filter del RF perché consenta solo il traffico strettamente necessario ai vari script.

redir.sh [RF, 12] – Questo script si pone in un ciclo infinito nel quale, ad ogni iterazione:

- interroga in pochissimi secondi tutti i 59 server via SNMP, per rilevare su ognuno se il demone *cron* è in esecuzione
- riconfigura il packet filter perché, per ognuno dei 250 client, il traffico ssh verso il RF sia autorizzato e reindirizzato verso un server scelto casualmente tra quelli che hanno *crond* attivo (si ricordi che la shell fornisce la variabile $RANDOM)

Indicare nei commenti come configurare l'agent SNMP dei server per consentire il controllo

script.sh [server, 11] – Lo script lancia in sequenza tutti gli script presenti in `~/jobs/` misurandone il tempo di esecuzione e al termine di ogni comando eseguito va ad aggiungere a LDAP una entry di classe *exec* coi relativi dati. Questo script è lanciato ogni ora da cron; per evitare sovrapposizioni tra esecuzioni, deve automaticamente terminarsi se il suo tempo totale di esecuzione supera i 50 minuti, garantendo anche la terminazione dello script correntemente lanciato da `~/jobs/`