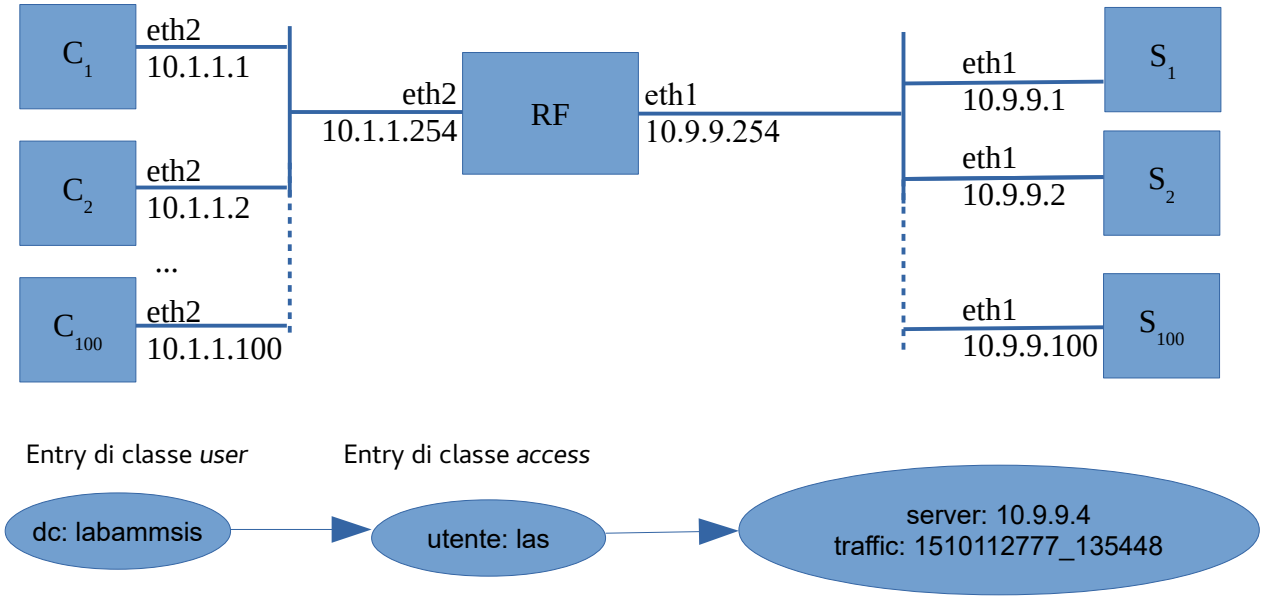
Laboratorio di Amministrazione di Sistemi T

Prova pratica del 10 novembre 2017

Descrizione generale del problema



Il sistema rappresentato in figura prevede un insieme di 100 Client, 100 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.
Ogni richiesta viene inviata via logging remoto, e il RF ricava via SNMP l'identità dell'utente che l'ha prodotta, per verificare che provenga da un utente abilitato, registrato su di una directory LDAP installata su RF stesso.
Nella directory gli utenti sono elencati in un primo livello di entry, e le connessioni a loro consentite sono rappresentate da un secondo livello che identifica i server raggiungibili e registra il traffico effettuato per ogni connessione autorizzata.

File da consegnare
(tra parentesi quadre la collocazione, non fa parte del nome)

access.schema.ldif – Definire i tipi di attributo *utente*, *server*, *traffic* (stringhe) e le classi *users* e *access* che li utilizzino come in figura: le entry di classe *users* hanno solo l'attributo utente, le entry di classe *access* hanno l'attributo *server* e possono avere attributi *traffic* che contengono una stringa nel formato *timestamp_traffico*.
Ad esempio, l'esistenza delle entry in figura stabilisce che l'utente *las* ha il diritto di collegarsi al server *10.9.9.4* e che all'istante 1510112777 è terminata una connessione (dall'utente al server) che ha generato complessivamente 135448 byte di traffico.

exec.sh [client] – Lo script accetta come parametri un IP di un server e un nome (assoluto) di un programma disponibile sul client medesimo in cui viene lanciato lo script, ed è usato da un qualsiasi utente di un client per chiedere l'esecuzione remota del programma sul server. Se lo script rileva che un altro utente ha un'istanza di *exec.sh* in esecuzione, termina con un messaggio d'errore, altrimenti

- attraverso rsyslog, fa comparire nel file **`/var/log/req.log`** del RF una riga con l'IP del client e quello del server;
- si pone in attesa che il RF inizi a rispondere al *ping*;
- se la risposta non arriva entro 10 secondi, lo script termina segnalando l'errore;
- se la risposta arriva entro 10 secondi, confermando l'accettazione della richiesta, lo script fa si che il programma indicato venga eseguito sul server remoto, e che output ed errori generati vengano salvati in locale sul client nei file **`~/out`** e **`~/err`**

firewall.sh [router] – accetta come parametri un comando e gli ip di un client e di un server. Se il comando è **`open`**, autorizza sul packet filter il traffico necessario per consentire a *exec.sh* del client di eseguire il programma sul server, e sblocca sempre sul packet filter la possibilità di rispondere ai *ping* del client; se il comando è **`close`** elimina le regole inserite con **`open`**

auth.sh [router] – Lo script esamina senza mai interrompersi il file **`/var/log/req.log`** e per ogni richiesta che appare:

- interroga via SNMP il client che l'ha originata per ricavare l'utente che ha in esecuzione *exec.sh*;
- interroga la propria directory LDAP per verificare se esiste una entry connessa all'utente per il server indicato nella richiesta;
- in caso positivo, invoca *firewall.sh* per autorizzare il traffico necessario a *exec.sh*
- configura *cron* per lanciare *timeout.sh* coi paramentri opportuni ogni 10 minuti, per gestire la terminazione della connessione in caso di inattività

timeout.sh [router] – Lo script necessita di 3 parametri: IP/utente del client e IP server relativi a una connessione autorizzata. Controlla il traffico rispetto all'ultima esecuzione (di se stesso), e se non ne rileva, rimuove per mezzo di *firewall.sh* le regole inserite da *auth.sh* e aggiorna la directory LDAP incrementando il contatore relativo a utente/server del numero di byte complessivamente osservanti durante tutta la durata della connessione, e provvede a riconfigurare *cron* per rimuovere la pianificazione della propria stessa esecuzione.

init.sh [router] – Lo script configura il packet filter del RF perché consenta inizialmente solo il traffico strettamente necessario al funzionamento dei vari script del sistema.

conf.txt [–] - Raccogliere in questo file tutte le modifiche/predisposizioni da apportare ai sistemi (es. configurazioni di servizi, generazione di dati preliminari, ecc.) necessari sulle varie macchine per consentire i comportamenti richiesti dagli script