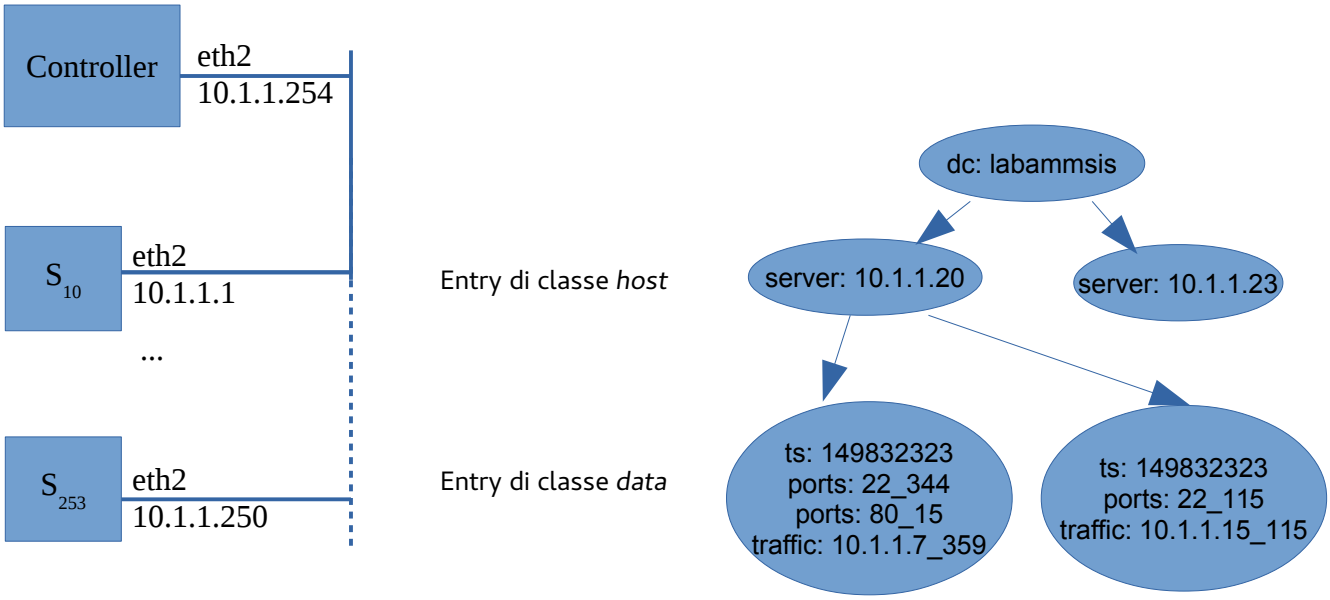
Laboratorio di Amministrazione di Sistemi T

Prova pratica del 13 settembre 2017

Descrizione generale del problema



Il sistema di monitoraggio rappresentato in figura prevede un insieme di 250 Server e un Controller. Il Controller verifica lo stato di attività dei server, garantendo che quando sono accesi essi si trovino in una di due situazioni: o sono in grado di inviargli informazioni sul traffico che ricevono, oppure devono essere isolati dal resto dei server per evitare intrusioni non rilevabili.
Quando un server si spegne, il controller deve memorizzare sulla propria directory LDAP un resoconto del traffico che esso ha ricevuto.

File da consegnare

(tutti gli script e i file di configurazione sono collocati sul Controller salvo eccezioni esplicitamente indicate)

getdata.sh – Lo script termina appena avviato se trova un file di nome `/tmp/getdata.complete`. Altrimenti, esegue una scansione dell'intervallo di indirizzi dei server per rilevare via SNMP quali sono accesi (la mancata risposta viene considerata sintomo di server spento) e tra questi quali hanno il processo `rsyslogd` in ascolto. Importante: implementare il controllo in modo che l'intero set di tutte le potenziali risposte sia ricevuto nel giro di pochissimi secondi.

Lo script produce su disco tre elenchi (in senso astratto - si scelga la forma di memorizzazione che si ritiene più adatta): uno con gli indirizzi dei server spenti, uno con gli indirizzi dei server accesi ma con `rsyslogd` inattivo, e uno con gli indirizzi dei server su cui `rsyslogd` è attivo.
Al termine, crea il file `/tmp/getdata.complete`
Indicare nei commenti:

- come configurare l'agent SNMP dei server per consentire tale verifica
- come far eseguire automaticamente ogni minuto lo script

compare.sh – Lo script controlla continuamente se esiste il file `/tmp/getdata.complete`. Ogni volta che lo rileva, avvia la procedura seguente:

- Per ogni server acceso con `rsyslogd` spento, vi si collega e vi lancia *activate.sh*
- Per ogni server acceso con `rsyslogd` attivo, lancia localmente al Controller lo script *collect.sh <server_ip>* in background (se non ne è già in esecuzione un'istanza)
- Per ogni server spento per il quale esiste un'istanza di *collect.sh* lanciata in precedenza, manda a quest'ultima un segnale USR1
- Cancella il file `/tmp/getdata.complete`

activate.sh (installato sui server) – Lo script configura `rsyslogd` per inoltrare al Controller i messaggi ricevuti dal packet filter, e tenta di avviare il demone `rsyslogd`. Se rileva che il demone si è avviato, configura il packet filter in modo che mandi al logger di sistema tutti i pacchetti ricevuti dagli altri server; altrimenti, configura il packet filter per impedire tutto il traffico in ingresso e uscita dal server, con l'unica esclusione del traffico relativo alle connessioni SSH e SNMP provenienti dal Controller, che devono funzionare correttamente.

collect.sh – Lo script legge senza interruzione il file `/var/log/pacchetti.log` (indicare nei commenti come configurare rsyslogd sul Controller perché in esso confluiscano i messaggi inviati dai packet filter dei Server). Prende in considerazione i soli messaggi inviati dal server specificato come parametro, e mantiene aggiornati due set di contatori:

- per ogni porta, il numero di pacchetti visti
- per ogni server, il numero di byte complessivamente ricevuti

Quando lo script riceve il segnale USR1, crea una entry LDAP di classe *data* (al di sotto della entry di classe *host* corrispondente al server specificato come parametro), inserendo in *ts* una marca temporale dell'istante di creazione e negli attributi *ports* e *traffic* i dati raccolti fino a quel momento, sotto forma di stringhe del tipo *<entità>_<contatore>* (ad esempio, se ha rilevato 15 pacchetti per la porta 80, inserirà `ports: 80_15`, se ha rilevato 6500 bytes dall'host 10.1.1.7, inserirà `traffic: 10.1.1.7_6500`)

tr.schema.ldif – Definire i tipi di attributo `ts` (intero), `server, ports, traffic` (stringhe) e le classi `host` e `data` che li contengano come esemplificato in figura.