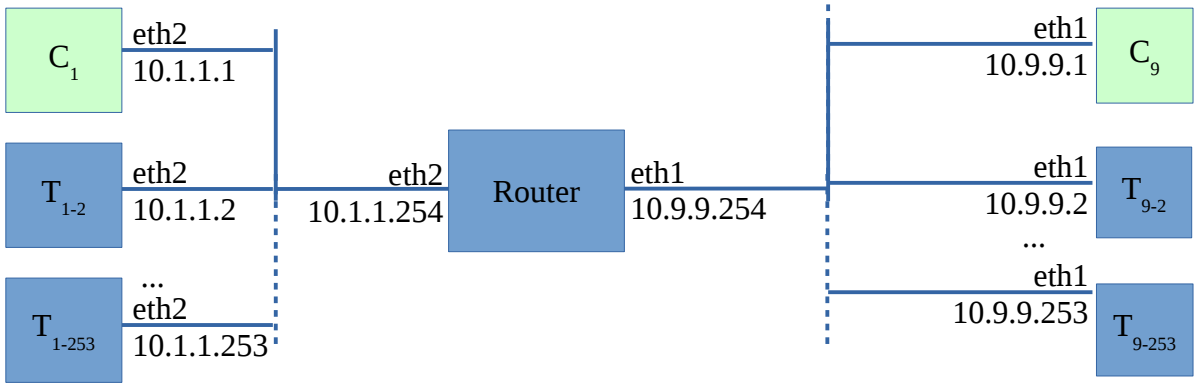
**Laboratorio di Amministrazione di Sistemi T**

# Prova pratica del 13 luglio 2016

## Descrizione generale del problema




Un sistema anti-intrusione distribuito e ad alta disponibilità è costituito da una coppia di macchine “controller” (10.1.1.1 e 10.9.9.1) che in condizioni normali tengono sotto controllo le macchine “target” collocate sulle rispettive reti locali, segnalando al router gli IP da bloccare perché generano traffico illecito (definito secondo pattern memorizzati su di una directory LDAP ospitata dal router stesso).

In caso di guasto di uno dei due controller, il router può fare in modo che l'altro ne prenda il posto, monitorando le macchine della rete remota oltre a quelle della rete locale (si assuma come ipotesi che la due macchine non si guastino mai contemporaneamente).

File da consegnare:

**pattern.schema.ldif** – Definire gli attributi LDAP *ip* e *badstring* (testuali) e la classe *blacklist* che li preveda obbligatoriamente entrambi. Le entry di questa classe conservano nei valori multipli dell'attributo *badstring* stringhe che rappresentano pattern di attacchi pericolosi per la macchina *ip*.

**detect.sh** (gira sul router) **–** Osserva senza interruzione tutto il traffico TCP tra le due reti, limitatamente ai primi 100 byte di ogni pacchetto. Conta tutti le stringhe di payload che passano. Quando rileva che un payload è stato catturato più di 10 volte, smette di contarlo e lo inserisce tra le *badstring* di tutte le entry LDAP già esistenti di classe *blacklist*.

**dump.sh** (gira sui target) **–** Effettua senza interruzione un dump su standard output di tutto il traffico TCP in ingresso alla macchina, ad eccezione di quello proveniente dalla rete locale (es. per il target 10.1.1.2 non va catturato il traffico proveniente dalla rete 10.1.1.0/24), in modo che i parametri di rete del pacchetto e tutto il suo payload in formato testuale siano disposti su di un'unica riga (suggerimento: si noti che con il comando adatto a questo esercizio, ogni nuovo pacchetto è marcato da una riga che inizia con timestamp e "IP", che può essese omessa).

**filter.sh** (gira sui target) – Usando *dump.sh* osserva il traffico, e se una riga fa match con almeno uno dei pattern considerati pericolosi per il target su cui gira, la coppia di indirizzi sorgente/destinazione in essa contenuta deve essere inviata alla facility *local1.alert*.
Lo script deve caricare i pattern pericolosi dal router, via LDAP, alla partenza, e ricaricarli ogni volta che riceve il segnale SIGHUP.
Indicare nei commenti: come vanno configurati i demoni rsyslog dei target e dei controller perché i messaggi etichettati *local1.alert* prodotti da qualsiasi target vengano scritti sul file `/var/log/anomalies` di entrambi i controller.

**check.sh** (gira sui controller) – Legge senza mai interrompersi le righe che compaiono nel file `/var/log/anomalies` e mantiene un set di contatori che tengono traccia di quante volte viene osservata ogni coppia di IP sorgente/destinazione.
Quando un contatore supera un valore di soglia (passato come parametro sulla riga di comando), lo script, impostando i parametri opportuni, lancia *firewall.sh* sul router per bloccare il traffico tra la corrispondente coppia di IP.
Indicare nei commenti: come configurare controller e router per consentire la connessione senza ostacoli

**firewall.sh** (gira sul router) **–** Riceve sulla riga di comando una coppia di IP. Se il packet filter contiene già regole riguardanti l'inoltro di traffico tra di essi, non fa nulla, altrimenti inserisce regole per bloccare l'inoltro di traffico tra di essi, e pianifica la modifica automatica di tale regola dopo 15 minuti, in modo che venga rimosso il blocco e al suo posto iniziato un logging del traffico scambiato tra i due IP.

**failover.sh** (gira sul router) – Controlla via SNMP se i controller sono attivi (cioè funzionanti e col processo *check.sh* in esecuzione). Se entrambi i controller sono attivi, si accerta che sia presente una regola che evita l'inutile propagazione da una rete all'altra dei messaggi di log generati dai target, mentre se rileva che un controller non è attivo, si accerta che i messaggi dei target collocati sulla rete in cui il controller è guasto possano raggiungere l'altro controller.
Indicare nei commenti:

- come garantire che lo script sia eseguito automaticamente ogni 5 minuti
- come configurare l'agent SNMP dei controller per consentire la verifica