SNMPScan

Progetto Gestione di Reti anno 2016/17

Marco Cameriero

`SNMPScan` è un network scanner che ricerca agent SNMP.

1 Introduzione

SNMP é un protocollo relativamente semplice e leggero per la configurazione, la gestione ed il monitoraggio degli apparati di rete. É un protocollo di livello applicativo che utilizza la porta UDP/161.

`SNMPScan` scansiona una determinata subnet alla ricerca di agent SNMP gestibili. L'applicazione effettua una scansione dell'intera subnet, inviando ad ogni singolo host una *Get* SNMP richiedendo il valore di `system.sysName.0`, che contiene il nome dell'agent. Se l'agent risponde positivamente, viene considerato attivo, ed il suo nome ed il suo indirizzo verranno stampati a schermo al termine dell'operazione.

2 Implementazione e struttura interna

`SNMPScan` utilizza la libreria `libsnmp`, parte del pacchetto `net-snmp`, per la comunicazione con gli agent SNMP. La libreria si occupa dell'implementazione del protocollo SNMP e della comunicazione con i singoli agent.

La logica principale dell'applicazione é all'interno della funzione `do_scan`, la quale si aspetta un puntatore ad una struttura `struct scan_state` ed il massimo grado di parallelismo (quante richieste contemporanee effettuare al massimo). All'interno della struttura `scan_state` é contenuto tutto lo stato necessario per la scansione attuale, tra cui la subnet da visitare, gli host trovati attivi e le richieste ancora pendenti. Le liste sono semplici linked list implementate con delle macro in `list.h`.

Il ciclo principale effettua quante piú richieste possibili (entro il limite impostato dall'utente) e le memorizza in una coda. Dopodichè si sospende in attesa che almeno una di queste abbia successo o vada in timeout o errore. Al ritorno dalla `select`, la funzione `snmp_read` o `snmp_timeout` invoca il callback delle singole sessioni per le quali é disponibile un aggiornamento di stato (errore o risposta). Notare come il codice sia asincrono, ma comunque single thread. La funzione di callback `on_snmp_response`, a questo punto, aggiunge l'host alla

lista di host attivi in caso di risposta positiva specificando anche il valore dell'oid richiesto `system.sysName.0`, oppure lo ignora completamente in caso di errore.

Terminata la scansione, stampa la lista di tutti gli host attivi con i relativi indirizzi e nomi.

3 Test ed utilizzo

`SNMPScan` supporta le seguenti opzioni:

- `-h`: Visualizza un aiuto veloce sulle opzioni.
- `-c`: Community SNMP da utilizzare per le richieste. Il default é `public`.
- `-r`: Numero di ritrasmissioni da effettuare prima di considerare un host down. Il default é `1`.
- `-t`: Timeout (in millisecondi) da attendere per ogni richiesta SNMP. Il default é `400ms`.
- `-p num`: Limita il massimo numero di richieste contemporanee. Il default é `4`.
- `-d[d]`: Abilita i log per il debugging. Utilizzare `-dd` per un output piú verboso.

Lo script `test.sh` puó essere utilizzato per testare velocemente `SNMPScan`: lo script avvia 6 container Docker con un agent SNMP ed esegue una scansione sulla subnet dei container.

```
$ ./out/snmpscan 172.17.0.0/24

Scan results:
- Total hosts scanned: 255
- Hosts up: 6

Host 172.17.0.2:
    SNMPv2-MIB::sysName.0 = STRING: 0e60fcd56f61

Host 172.17.0.3:
    SNMPv2-MIB::sysName.0 = STRING: 3d110d32aaea

Host 172.17.0.4:
    SNMPv2-MIB::sysName.0 = STRING: 0d4eea195074

Host 172.17.0.5:
    SNMPv2-MIB::sysName.0 = STRING: 1ccfbdad74ab
```

```
Host 172.17.0.6:
    SNMPv2-MIB::sysName.0 = STRING: 4639c3c64453

Host 172.17.0.7:
    SNMPv2-MIB::sysName.0 = STRING: e3319d4903d4
```