Network Devices Monitor

Brassat Alexandru[1]

Universitatea Alexandru Ioan Cuza Iași, Facultatea de Informatică
alexandru.brassat@gmail.com

Abstract. O platformă de monitorizare a unor surse de date de pe diverse mașini (agenți), cu posibilitatea de a extrage metrici și statistici din datele colectate.

1 Introducere

Implementati o platforma de monitorizare a operatiunilor de retea si a incidentelor de securitate. Sa se realizeze o solutie ce afiseaza centralizat, intr-un dashboard, metrici si statistici generate din log-urile colectate din echipamente de retea, endpoints si servere. Platforma va fi multiuser, multithreaded, cu un dashboard configurabl, capabila sa colecteze nativ syslog, modulara – cu posibilitatea de a adauga surse de date, agenti si filtre. Implementarea presupune configurarea unei infrastructuri fizice sau virtuale cu echipamente active si/sau endpoints, pe care pentru testare se va putea instala software third party (agenti).
Proiectul constă în trei aplicații: Client, Server și Agent.

2 Tehnologii utilizate

- **Socket-uri TCP** - Folosite în cadrul comunicării Server - Agent, pentru a asigura recepționarea tuturor datelor, în aceeași ordine în care au fost transmise
- **Conexiune de control** - Similar cu cea din FTP, folosită în cadrul Agentului pentru a asigura o comunicare ușoară cu serverul, în cadrul căreia se schimbă informații "organizatorice", de exemplu transmiterea datelor mașinii în momentul conectării la server, sau a comenzilor de configurare a Agentului(adăugarea de surse noi de date, validare, filtre, etc.)
- **Socket-uri UDP** - Folosite în cadrul comunicării Client - Server, fiindcă requesturile clientului nu depind de o sesiune anume de lucru, ele vor fi transmise prin UDP, recepționarea datelor fiind asigurată la nivelul aplicației, prin retransmiterea periodică a requestului. Fiecare request își alocă un port, acesta fiind închis în momentul primirii răspunsului. Datorită retransmiterii requestului, este posibil ca o copie a răspunsului de la server să polueze următoarea aplicație care va folosi același port.
- **mutex** - pentru sincronizarea accesului mai multor threaduri la resurse partajate. Este folosit în cadrul Agentului, pentru ca mai multe threaduri să poată transmite date folosind același socket.

3 Arhitectura Aplicației

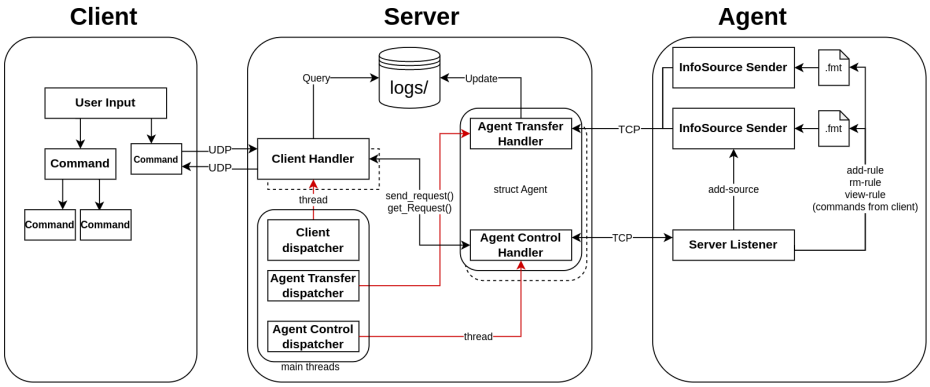

Fig. 1. Diagrama aplicației

3.1 Agenții

Agenții urmăresc datele scrise în niște fișiere log, numite "info sources" în cadrul aplicației. Fiecare fișier log are un fișier .fmt asociat, în care sunt specificate mai multe "reguli", adică formate de mesaje, cu parametrii de urmărit. În momentul în care fișierul log înregistrează un mesaj care respectă una din regulile căutate, agentul trimite serverului acel mesaj în format json, pentru a fi stocat în baza de date.

Agentul poate primi de asemenea comenzi de schimbare a formatului urmărit într-un fișier log (ștergere/ adăugare de formate de mesaje logate, vizualizarea acestora) sau de adăugare a unui nou fișier log. Aceste comenzi sunt procesate în mod iterativ.

3.2 Clientul

Aplicația client este una textuală, care se folosește de UDP pentru a transmite cereri serverului.

Lista de comenzi:

- list all agents
- run <file-name> as script in client terminal
- show information of <agent-name> (name, OS, etc)
- show active info sources of <agent-name>
- restart <agent-name>
- show number of rule pages (there are 10 rules/page)

- add a new log file to watch for <agent-name>
- show names of all rules on page <page>
- remove tracked rule in log file
- ask for entries matching <conditions> from the log database
- show actual rule referred to as <rule-name> in <agent-name>
- add new rule to track in <agent-name>
- graph the count of entries which match <conditions> in a date interval

3.3 Serverul

Serverul reține o listă cu Agenții conectați și thread-ul responsabil de fiecare Agent. În folderul "logs/ID-Agent/" se găsesc un fișier "agent.info", cu informații generale ale agentului, și, pentru fiecare sursă de date urmărită, un document în format json, aceasta fiind baza de date a serverului care poate fi interogată. Această bază de date conține pentru fiecare mesaj urmărit numele regulii pe care o respectă și lista parametrilor urmăriți, în tuple de tip de date, nume, valoare.

Nu există un mecanism de verificare a autenticității cererilor clienților, însă există o limită pe numărul de cereri client ce pot fi preluate la un moment dat, orice cerere venită după depășirea limitei fiind ignorată.

Comenzile clienților sunt preluate de un thread specializat, care așteaptă la un port prestabilit și generează noi thread-uri care să se ocupe cu executarea comenzilor. Prin urmare, serverul este unul concurent.

4 Detalii de implementare

4.1 Client

Clientul preia comenzi de la linia de comandă, pe care le trimite mai apoi serverului prin socket-uri UDP în formatul [cod comandă][parametrii despărțiți prin ”]. Mesajul este retransmis de un număr de ori, pentru a anticipa eventualele inconsistențe specifice UDP-ului.
Fiecare cerere folosește un socket pentru comunicarea cu serverul, ceea ce înseamnă că programul poate fi multi-threaded fără alte modificări, fiecare cerere fiind în mod unic identificată prin adresa ip și portul socketului de la care trimite. De exemplu, comanda "graph" trimite în paralel toate query-urile necesare pentru desenarea graficului.

4.2 Server

Serverul rulează pe trei thread-uri principale: thread-ul "client dispatcher", "agent transfer dispatcher" și "agent control dispatcher". Pe lângă acestea, thread-ul main verifică periodic căderea primelor trei și le repornește dacă e cazul

- "client dispatcher" - Primește datagrame de la clienți, iar dacă încă nu s-a depășit limita de cereri procesate în același timp, creează un thread nou care să trateze cererea.
  Protocolul folosit este UDP, fiindcă cererile nu depind de informații specifice unei conexiuni sau unei sesiuni de lucru, toate sunt egale din acest punct de vedere.
  - "client treat" - acest thread parsează comenzile primite și, în cazul în care este o comandă la adresa bazei de date sau a serverului(c-query, prop și list), o tratează pe loc. Altfel, comanda este adresată unui agent și este dată mai departe acestuia. Acest tip de comandă are mereu ca prim parametru numele agentului căruia îi este adresat. În acest caz, thread-ul trimite pe conexiunea de transfer a agentului comanda cu formatul [lungimea mesajului][cod comanda][thread id][restul parametrilor] si apoi așteaptă să fie notificat de primirea răspunsului de către "agent control listener" (prin folosirea structurii Request si a functiei get_Request). Dacă așteaptă prea mult se "dezabonează" și se închide, pentru a nu ocupa resurse. Clientul este responsabil pentru reîncercarea comenzii.
- "agent control dispatcher" - acceptă conexiunile agenților la un port cunoscut, apelează fnc_agent_creator într-un thread nou pentru a se ocupa cu stabilirea conexiunii cu agentul.
- "agent transfer dispatcher" - acceptă conexiunile agenților la un port cunoscut, dar creează thread-uri dedicate agentului doar atunci când acesta a fost înregistrat în lista cu agenți conectați (prin prezența thread-ului de control și a login-ului cu succes, lucru realizat în funcția fnc_agent_creator).

Fiecare agent are câte două thread-uri dedicate:

- "agent transfer listener" se ocupă exclusiv de umplerea bazei de date, citind mesaje de tipul [lungime mesaj][info source][mesaj în format json] și adăugându-le în baza de date.
- "agent control listener" trimite comenzi agentului și primește răspunsuri de tipul [lungime mesaj][cod comandă ack][thread id][mesaj], mesajul fiind transmis mai departe thread-ului cu id-ul specificat, dacă mai este în viață. Acest thread mai este responsabil cu menținerea conexiunii, prin primirea unor mesaje cu heartbeat periodice, care resetează timer-ul de time out al agentului (time out în 10s, heartbeat în 5s).
  De asemenea, thread-ul acesta este cel care este conectat primul, cel prin care se face operația de login a agentului(prin transmiterea proprietăților agentului).

4.3 Agent

Agentul rulează pe mai multe threaduri, unul fiind dedicat pentru conexiunea de control, iar restul vor fi asociate câte unei surse de informație. La deschiderea aplicației, se inițializează automat sursa de informații "/var/log/syslog". Conexiunea de transfer este deschisă la inițializare, și este protejată printr-un mutex, care asigură trimiterea pe rând a mesajelor din diferite surse de informații.

5 Concluzii

Comunicare dintre Server și Agent ar putea fi îmbunătățită prin introducerea a două tipuri de mesaje:

- Mesaje critice, transmise prin TCP pentru a asigura primirea lor sigură și cât mai rapidă
- Mesaje uzuale, transmise prin UDP, cu validarea periodică a datelor stocate pe server cu cele de pe mașină

Implementarea unei interfețe grafice.

Folosirea unui Model-View-Controller în client: Reținerea răspunsului de la server într-un fișier temporar pentru fiecare widget, astfel încât widgeturi duplicate să nu ceară aceeași informație de mai multe ori.

În aceeași idee, se poate implementa în server un cache cu ultimele requesturi primite de la clienți, pentru a nu efectua mai multe operații costisitoare decât este nevoie(ex.: căutarea tuturor înregistrărilor din baza de date care au o anumită proprietate, sau validarea explicită a datelor de pe server).

"Smart queries" în client: pentru widgeturi care fac apel la aceleași informații sursă (ex.: redarea de loguri "raw" și afișarea unor statistici despre acestea), se poate cere de la server doar informația cu volumul cel mai mare, iar restul widgeturilor care se folosesc de o submulțime a acestei informații să o proceseze pentru a-și obține submulțimea relevantă. O idee de implementare ar fi reținerea listei de widgeturi active, iar în momentul în care este necesară solicitarea unei informații de la server, se caută în acea listă widgeturile compatibile pentru un smart query.

Un daemon în server care calculează periodic statistici generale des folosite și le reține într-un cache.

Adăugarea unui mecanism de verificare a autenticității cererilor clienților, prin instanțierea unei parole globale în server, clienții fiind nevoiți să transmită această cheie la fiecare mesaj trimis.

Suport pentru urmărirea unor tranzacții care cuprind mai multe mesaje logate.

References

1. https://profs.info.uaic.ro/ computernetworks/cursullaboratorul.php
2. https://github.com/KebabRonin/ProiectRetele