Reti di Calcolatori – 13 gennaio 2016

Si progetti un'applicazione Client/Server che, utilizzando le socket, aiuti un ingegnere informatico a identificare le più sicure alternative disponibili sul mercato per un determinato componente software (libreria, middleware, ecc.) da utilizzare nell'applicazione che sta progettando. Più precisamente, l'ingegnere è interessato a trovare il progetto software per cui è stato riportato il minor numero di vulnerabilità di sicurezza. L'applicazione deve presentare la seguente interfaccia:

find_safest_component server porta

dove ***server*** rappresenta il nome logico del Server e ***porta*** rappresenta il numero di porta del servizio.

Per prima cosa, il Client si deve interfacciare con l'utente, da cui riceve (via terminale) *il nome del database di vulnerabilità* (es., "NVD", "CERT", "OSVDB", ecc.), il *tipo del componente software* (es., "Libreria TLS", "Server Web", "Server E-mail", ecc.), il *sistema operativo* (es. "Linux", "FreeBSD", "Windows", ecc.) di interesse. Il Client deve quindi trasmettere le informazioni al Server, che a sua volta dovrà reperire le informazioni di interesse, elencandole in ordine di numero di vulnerabilità decrescente, e restituirle al Client.

A questo proposito, si supponga che le informazioni dei principali database di vulnerabilità siano salvate sul Server in altrettanti file di testo nella directory */var/local/vulnerability_databases/* (quindi, per esempio, le informazioni del database NVD saranno salvate nel file */var/local/vulnerability_databases/NVD.txt*, ecc.) Ciascuna riga di tali file conterrà le informazioni per uno specifico componente software, con (in quest'ordine) il numero di vulnerabilità riportate, il nome del componente software, il numero di versione del componente software, il tipo del componente software, il sistema operativo, ecc. (A ogni componente software possono corrispondere più righe, in quanto ci possono essere vari componenti che svolgono la stessa funzione e più versioni dello stesso componente, anche per sistemi operativi diversi.)

Una volta ricevute le informazioni dal Server, il Client le stampa a video e si mette in attesa della richiesta successiva. Il Client deve terminare quando l'utente digita "fine".

ATTENZIONE: Si realizzino il Client e il Server in C, ma il Client deve essere realizzato anche in Java.