Protocollo FTP
-
Progetto del corso di Reti di Calcolatori
A.A.: 2019-2020

Federico Cervino (mat. 923330)

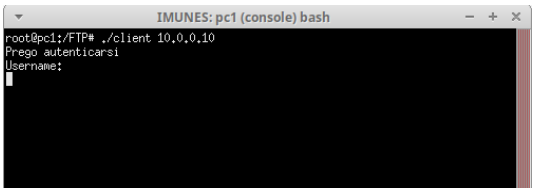
Indice dei contenuti

1. Introduzione

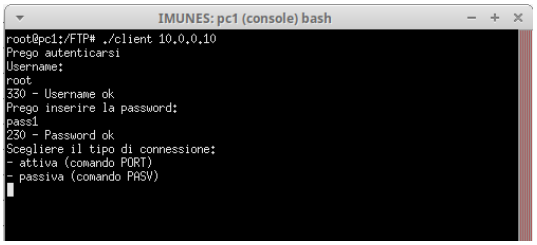
Il File Transfer Protocol è un protocollo del livello applicativo che permette di scambiare file tra due macchine, un server e un client, tramite una connessione TCP.
Per questo progetto ne è stata realizzata un'implementazione per sistemi operativi Linux tramite linguaggio di programmazione C sfruttando i meccanismi delle socket per la comunicazione tra i due nodi.
Il tutto infine è stato testato su una rete simulata tramite il software Imunes.

2. Interfacce

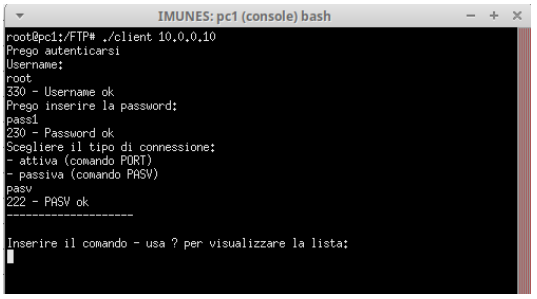
Le interfacce sono a riga di comando:



Interfaccia di autenticazione



Scelta del tipo di connessione



Inserimento dei comandi desiderati

3. Funzionamento

Il protocollo FTP si basa su due canali di comunicazione, uno su cui viaggiano i parametri che regolano la connessione e l'altro su cui viaggiano i dati veri e propri.
Anche questa implementazione del protocollo rispetta questo standard: infatti all'avvio dell'eseguibile lato client, con il comando *./client <indirizzo IP del server>*, viene creata la prima connessione di controllo col server (che si suppone costantemente attivo e quindi in ascolto in questo caso sulla porta 2021).
Solo in un secondo momento poi, quando sarà il momento di trasferire i dati veri e propri verrà creata anche la connessione dati.
Questa implementazione ammette entrambi i tipi di connessione: passiva (comando *PASV*), in cui è il server a decidere la porta a cui si dovrà collegare il client, e attiva (comando *PORT*), in cui invece è il server a connettersi alla porta specificata dal client.
Le socket sono basate su connessioni TCP come indicato dagli argomenti SOCK_STREAM presenti nelle funzioni di creazione dei descrittori delle socket e sono gestite dalle funzioni della libreria <sys/socket.h>. In particolare, per invio e ricezione di dati, vengono usate rispettivamente le funzioni send() e recv(). Anche per lo scambio di file vengono usate le stesse funzioni: di qualsiasi file infatti viene acquisita la dimensione, che viene usata per allocare lo spazio necessario nella memoria del programma tramite la funzione malloc(); in quell'area di memoria poi verrà caricato l'intero file tramite la funzione fread().
Il server inoltre gestisce i client generando nuovi processi di se stesso tramite la funzione fork(), è quindi capace di rispondere alle richieste di 5 client contemporaneamente.
Per quanto riguarda le autenticazioni possono avvenire anche in modalità anonima, con la quale non viene chiesta la propria password ma solo un indirizzo email da inserire in un file che registra gli utenti anonimi che hanno eseguito l'accesso.
I file scaricati infine sono salvati in una cartella creata dal programma chiamata *Downloads*.

4. Comandi supportati

I comandi supportati da questa implementazione dell'FTP sono 4, oltre ai due citati in precedenza (*PASV* e *PORT*):

- *?*: Visualizza la lista dei comandi.
- *LS*: Visualizza la lista dei file che il server può inviare ai client; ovviamente si limita a mostrare solamente i file voluti dal server, contenuti nella cartella *Files*.
- *ADDUSER*: Permette di aggiungere un utente alla lista se non vi è già presente, è però eseguibile solamente dall'utente amministratore (che in questo caso si chiama *admin*).
- *RETR*: Permette di scaricare il file desiderato.

5. Sicurezza

A differenza del protocollo base, questa implementazione dell'FTP prevede delle piccole accortezze volte a migliorarne la sicurezza, intesa come riservatezza e confidenzialità dei dati scambiati, riservate agli utenti autenticati e non agli anonimi.
Per prima cosa le password non viaggiano mai in chiaro lungo il canale di controllo, né durante l'autenticazione né durante l'esecuzione di *ADDUSER*.
Per quanto riguarda l'autenticazione, essa avviene tramite nonce inviato dal server che autenticherà l'utente solo se egli riesce a criptarlo correttamente tramite una funzione di hashing (in particolare MD5) usando come chiave la propria password. Questa modalità di autenticazione risulta efficace per contrastare gli attacchi di tipo replay attack, dato che il nonce cambia ad ogni richiesta di accesso.
Nel caso invece della password del nuovo utente aggiunto tramite *ADDUSER*, essa viene criptata con la password dell'utente corrente tramite XOR, e la stessa tecnica viene usata per criptare i risultati degli altri tre comandi.
Una password di almeno 10 caratteri alfanumerici usata come chiave infatti risulta efficace per esempio nel caso di un attacco a forza bruta, dato che genera un numero di possibilità pari a $8{,}39*10^{17}$ e un computer di media capacità computazionale impiegherebbe più di 2600 anni a eseguire un attacco del genere (contando una media di 10 milioni di tentativi al secondo), escludendo quindi anche la possibilità di effettuare attacchi di tipo man in the middle in quanto l'attaccante in questione non riuscirebbe a decifrare ciò le comunicazioni che intercetta.

6. Conclusioni

Dato lo scopo didattico di questo progetto per l'autenticazione è stata usata la funzione crypt() in modo che lavori con password al massimo di 8 caratteri e gli utenti autenticati anonimamente hanno accesso agli stessi dati a cui hanno accesso in maniera ''protetta'' gli altri utenti. Questi due fattori sono gli anelli deboli del sistema di riservatezza e confidenzialità studiato per questa implementazione di FTP e andrebbero corretti riservando aree di memoria per gli utenti autenticati e usando funzioni di cifratura che accettino chiavi da almeno 10 caratteri.
Esiste inoltre un limite alla dimensione dei file che possono essere trasferiti: la funzione usata per calcolare questo parametro è stat() che ritorna un valore di tipo off_t, interpretabile secondo il *GNU C Library Reference Manual* come un numero intero con segno che quindi può assumere il valore massimo di +2.147.483.647.
Da qui si ha che il valore massimo della dimensione di un file trasferibile è di 2GB.