Laboratorio di sistemi – classe 5^{a} – data 9/1/2020

Login “sicuro” - versione base

SI vuole implementare la fase di autenticazione utente (login) in una comunicazione client-server basata sul protocollo TCP. Si suppone che nel server siano già memorizzati gli account degli utenti (in un database, un file di testo, un file XML, o in memoria); l’obiettivo è implementare un protocollo che consenta agli utenti di autenticarsi senza che la password possa essere intercettata durante la comunicazione.

Requisiti

- Il server mantiene una copia in chiaro delle password degli utenti. (Su file di testo, file XML, database, etc.)
- Si richiede di implementare il protocollo con il solo ausilio di un algoritmo di *hashing* (SHA, per esempio).
- Client e server **non** devono scambiarsi la password dell’utente, né in chiaro, né in forma di *digest*.

Implementazione del protocollo: sfida→risposta

Per evitare di comunicare la password si può usare il metodo chiamato “**sfida→risposta**”, applicabile quando client e server condividono un *secreto* (la password del client). Il protocollo prevede il seguente scambio tra client e server :

1. *il client chiede di autenticarsi (eventualmente inviando subito il nome dell’utente)*
2. *il server invia al client una **sequenza casuale di caratteri***
3. *con la sequenza ricevuta dal server, il client crea un **ticket**, generando un digest della combinazione (**password+sequenza**); quindi lo invia al server.*
4. *Il server ripete la stessa operazione con la password del client recuperata dal proprio database e produce a sua volta un **ticket**, lo confronta con quello ricevuto dal client e verifica così le credenziali dell’utente.*

Nota bene: la precedente descrizione non specifica i dettagli del protocollo:

- Tipo e formato dei messaggi scambiati.
- L’algoritmo di *hashing* utilizzato.
- La lunghezza delle sequenza casuale di caratteri, né il modo con il quale viene combinata con la password (modalità che deve essere condivisa tra client e server).

Note sull’implementazione

Gestione account

Lato server si dovrebbe implementare una classe, `AccountStore`, con la funzione di gestire gli account (nome utente e password) degli utenti. Inizialmente, si può immaginare di creare e gestire i dati in memoria; successivamente sarà possibile modificare la classe in modo che recuperi i dati da uno *storage*.

Protocollo di comunicazione client-server

Occorre stabilire un protocollo condiviso da client e server. In generale: ad ogni messaggio del client, il server può rispondere con un errore se il messaggio non rispetta determinati vincoli. All'opposto, si può supporre che il server risponda sempre in modo corretto. Il protocollo deve stabilire:

- Il formato dei messaggi.
- Tutti i possibili messaggi, compresi i possibili errori restituiti dal server.

Segue un esempio che può essere utilizzato come spunto (oppure può essere implementato così com'è). A sinistra è specificato il formato dei messaggi inviati dal client, a destra il formato delle risposte del server (con alcuni esempi):

CLIENT	←→	SERVER
`<richiesta> <corpo richiesta><nl>`		`<codice-risposta> <corpo risposta><nl>`
`LOGIN <utente><nl>`		**`0 <sequenza casuale><nl>`** **`1 Utente non trovato<nl>`**
Avvia la richiesta di login, specificando l'utente da autenticare. Il server risponde con una sequenza casuale di caratteri, oppure con un errore se la richiesta è sbagliata o il nome utente non corrisponde a nessun account.		

`LOGIN paolo.rossi<nl>`
`LOGGIN` `paolo.rossi<nl>`
`LOGIN` **`paolo.rosssi`**`<nl>`
		`0 A14EFGhh56678h1A<nl>`
		`10 Richiesta non valida<nl>`
		`20 Utente non trovato<nl>`

`SECRET <hash(password+sequenza casuale)>`		**`0 Autenticazione riuscita<nl>`** **`30 Password non valida<nl>`**
Il client invia il *ticket* (hash di password+sequenza casuale). Il server verifica la corrispondenza con il proprio ticket e risponde conseguentemente.		

Nota bene, nell'esempio:

- Ogni messaggio è terminato da un fine riga: `<nl>`.
- La risposta del server è prefissata da un codice numerico. 0 indica il successo dell'operazione.