ESEMPIO:

Su un pc la query injection funziona su un altro no. capire perché.

Le pagine consentono di inizialmente di scegliere la tipologia di utente (titolare o cliente). (Puoi utilizzare il database negozio\_nuovo se non vuoi ricrearlo)

Mappa del sito

O Menu.html

Per la connessione al database utilizzano il file ConfigConnTitolare.php che consente l’accesso con i privilegi del titolare

* Logintitolare.php (un: titol1, pw:123)
  + Inserisci\_nuovo\_prodotto.php
  + Visualizza\_prodotti.php
  + Visualizza\_ordini.php
  + Visualizza\_clienti.php
* Logincliente.php (uso il file logincliente\_soggetto\_SQL\_INJECTION.php) (un: gianni, pw:123)
  + Visualizza\_prodotti
  + Visualizza\_miei\_ordini
  + Effettua\_ordine

Per la connessione al database utilizzano il file ConfigConnCliente.php che consente l’accesso con i privilegi del cliente

Per visualizzare il problema i due file di connessione sono stati modificati rispetto a quelli realizzati in laboratorio. In particolare:

Il file logintitolare, per non controlla l’hash dell password con il metodo password\_verify() ma direttamente il valore della password memorizzato nella tabella. (questo solo perché così posso inserire dei titolari con pw direttamente nella tabella in phomyadmin)

Il file logincliente\_soggetto\_SQL\_INJECTION.php è stato in parte modificato rispetto a loginTitolare, in modo da essere soggetto a SQL INJECTION. (Utilizza il metodo multy\_query($query) anziché query($query)).

Il file “logincliente\_soggetto\_SQL\_INJECTION.php “controlla l’hash della password, quindi gli eventuali clienti inseriti direttamente nella tabella clienti in cui la password non è hashata, non possono accedere. Creare un utente direttamente dalla pagina web “register.php”

L’esempio mostra due cose:

1. Che per effettuare la connesione al database con diversi profili utente è suffciente creare un file ConfigConnessione per ciascuno dei profili desiderati.
2. Il login del cliente è vulnerabile al seguente attacco SQL INJECTION:

Inserendo la seguente stringa come username:

' OR 1=1; DELETE FROM prodotti WHERE id\_prodotto=3; --

Viene eliminato il prodotto con id\_prodotto=3 dalla tabella prodotti.

Poiché il login del cliente è vulnerabile alla SQL INJECTION, se la connessione al database avviene con i privilegi del cliente, il tentativo di SQL INJECTION non va a buon fine poiché il cliente non ha il privilegio di eliminare prodotti. Se invece la connessione avviene con privilegi maggiori (root oppure cliente) la SQL INJECTION va a buon fine.

Per verificare basta provare la SQL INJECTION

* Prima utilizzando il file di connessione **ConfigCliente.php** per il cliente, Si verifica un errore SQL ma il prodottocon id\_prodotto=3 **NON VIENE ELIMINATO**
* Poi utilizzando il file di connessione **ConfigTitolare.php** per il cliente (privilegi maggiori), Si verifica un errore SQL e il prodotto con id\_prodotto=3 **VIENE ELIMINATO**

Questo per dimostrare che la gestione corretta dei privilegi è uno dei modi (non l’unico) per aumentare la sicurezza dei dati memorizzati nel database. La sicurezza dei dati non si ottiene semplicemente con una soluzione ma con un insieme di soluzioni, ad esempio: connessioni con privilegi diversi, utilizzo di crittografia, utilizzo di opportune funzioni pk (mysqli->query() anziché mysqli->multiQuery()), ecc.