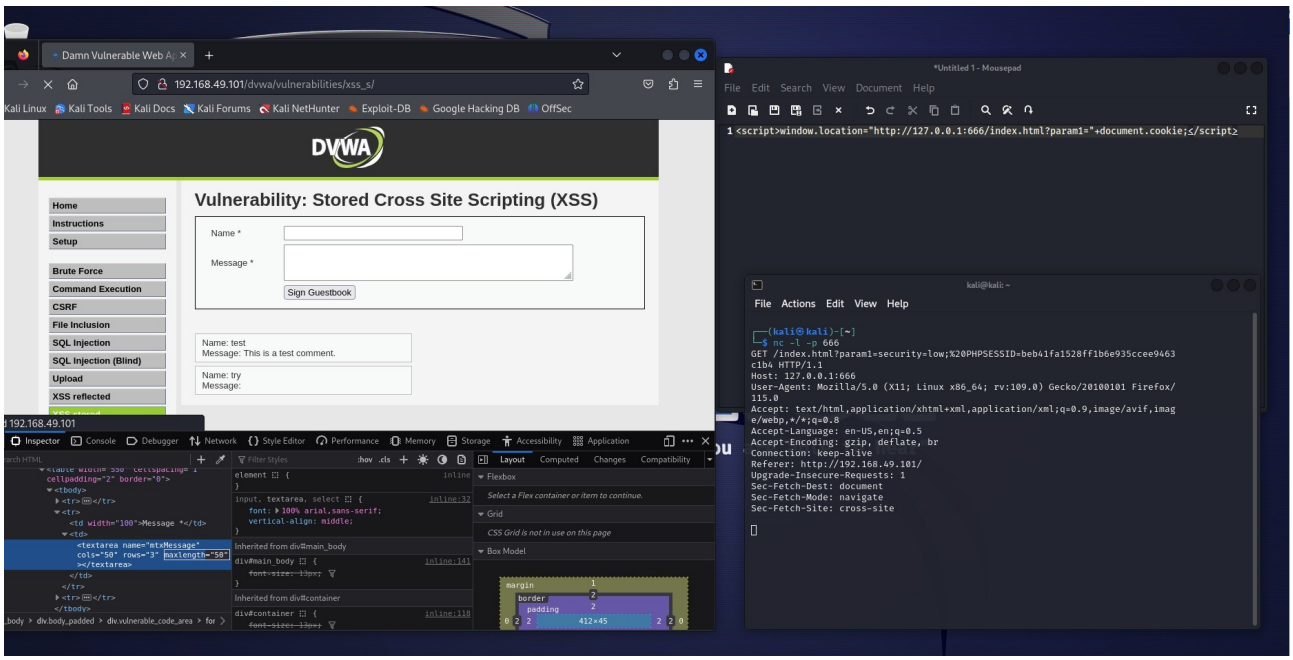
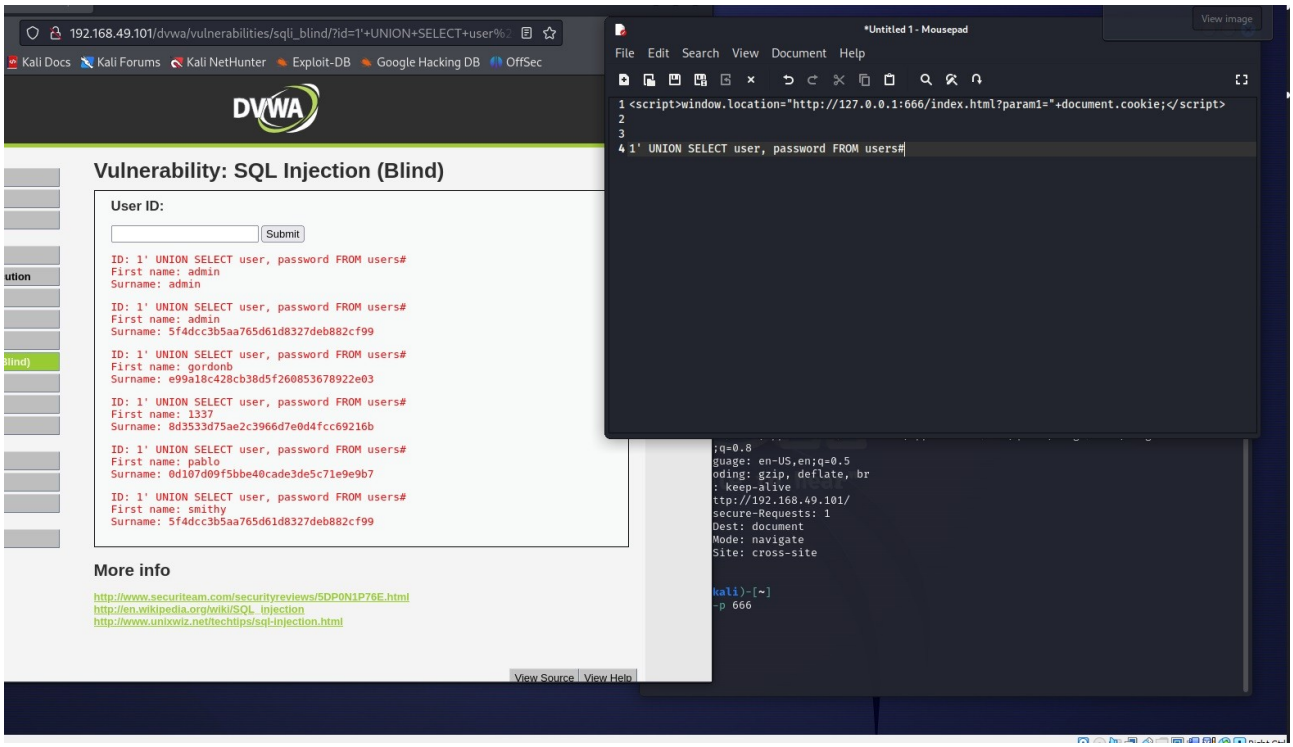


Come da immagine ho effettuato un attacco xss(cross site scripting) persistente; a differenza di quello riflesso fatto nella precedente esercitazione(vedi S6L2) che veicola il codice malevolo tramite la modifica dell'indirizzo della pagina, il persistente si effettua sulla pagina stessa in quanto il verbo della pagina e' di tipo POST, il che significa che si possono caricare all'interno del database i dati malevoli contaminando la pagina stessa e quindi chiunque entri in quella pagina viene attaccato. Attivandosi solo se la vittima utilizza l'url modificato, lo xss riflesso e' molto meno pericoloso di una pagina perennemente infetta come nello xss persistente.
Nell'immagine si puo' notare che l'ostacolo all'inserimento del codice e' la lunghezza massima di caratteri ammessi nel riquadro message visibile aprendo la modalita' sviluppatore sul browser; modificando il numero con una cifra piu' alta si puo' ovviare al problema; in questo caso ho inserito il numero 300 e poi mi e' stato possibile inserire il codice che ho scritto su mousepad.
Nel terminale mi sono messo in ascolto sulla porta 666(cosi' il demone si sente piu' a suo agio) dove si possono vedere i risultati ottenuti.



Come da immagine, ho eseguito una SQLinjection ma questa volta su una pagina che non stampa le risposte di errore e quindi risulta piu' difficile individuare una possibile vulnerabilita' in fase di assesment; per il resto la procedura e' stata identica all'esercizio S6L2, dove ho ampiamente approfondito tutte le varie fasi del processo, e che riporto a seguire

“Anche nel caso dell'SQL injection bisogna prima andare per tentativi; si comincia a scrivere diversi elementi come ad esempio comandi sql, operatori logici, commenti etc... e si vede come risponde la pagina.
In base a come risponde si costruisce la stringa che modifichi la query al database; si puo procedere in 2 modi :

- il primo e' usando l' operatore logico OR che consegna sempre un risultato TRUE in tutti i casi sia presente almeno un risultato TRUE e in questo modo l'intero parametro WHERE(presente nella parte di query non modificabile) viene annullato e di conseguenza compare l'intera tabella con tutte le tuple degli id
- il secondo e' usando il comando UNION, unendo il SELECT di first name e surname dell'id 1(il numero e' l'unica cosa scelta da noi in quanto tutto il codice scritto davanti a 1 e' fisso e non modificabile) con il SELECT di user e password di tutte le tuple prese dalla stessa tabella(questo SELECT e' completamente inventato da noi e modifica il senso di tutta la query non modificabile che c'e' davanti e veniamo a conoscenza degli attributi user e password presenti in tabella, facendo vari tentativi si e' scoperto il nome della tabella); alla fine si inserisce un cancelletto per rendere commento qualsiasi cosa scritta dopo.

Osservando il codice sorgente della pagina si puo notare come l'unica variabile modificabile nella query e' id perche' ha la $ davanti; noi abbiamo fatto le nostre aggiunte sostituendo quel singolo elemento al fine di stravolgere totalmente il risultato della query”