הגנה מפני – SQL Injection

שמות ותעודות זהות:

- מקסי ישראל קרוטינסקי: 208225664
- גל משה טאייב: 207338104

שימוש ב־ORM (Object-Relational Mapping)

ORM הוא כלי שמתרגם בין אובייקטים בתוכנה לבין טבלאות במסד הנתונים.
במקום לכתוב שאילתות SQL ידניות, מפתחים משתמשים באובייקטים (כמו טבלאות, רשומות ומאפיינים).
ORM בונה את השאילתות בצורה בטוחה כך שהקלט של המשתמש לא מוזן ישירות ל־SQL, ולכן מונע הזרקות זדוניות.
דוגמאות: Django ORM (Python), Hibernate (Java), Entity Framework (.NET).

הגבלת הרשאות למסד הנתונים(Least Privilege Principle)

העיקרון קובע שיש להעניק לכל רכיב במערכת אך ורק את ההרשאות שהוא באמת צריך.
במקרה שלנו – המשתמש שאיתו האפליקציה פונה למסד הנתונים צריך להיות מוגבל – למשל לאפשר רק קריאה (SELECT) וכתיבה (INSERT), ללא הרשאות מסוכנות כמו DROP או ALTER.
כך, גם אם תוקף מצליח לחדור עם SQL Injection, הוא לא יוכל לבצע פעולות הרסניות על המסד.

שימוש ב־WAF (Web Application Firewall)

WAF הוא חומת אש שממוקמת בין הגולשים לאפליקציה, ומבצעת סינון של כל הבקשות המגיעות לשרת.
המערכת יודעת לזהות ניסיונות תקיפה כמו SQL Injection לפי תבניות ידועות (לדוגמה OR 1=1), ולחסום אותן עוד לפני שהן מגיעות לקוד האפליקציה.
יתרון חשוב: גם אם המפתח שכח לבצע סינון נכון – ה־WAF יכול לשמש כשכבת הגנה נוספת.

מה עשינו בפועל על מנת לתקן את הSQL INJECTION?

בגרסה המקורית של הקוד חיברו את קלט המשתמש ישירות למחרוזת ה SQL, מה שגרם לפגיעות חמורה ל-SQL Injection.
תוקף היה יכול להכניס קוד זדוני במקום שם משתמש או סיסמה ולעקוף את מנגנון ההתחברות.
התיקון נעשה על ידי שימוש בפרמטרים בטוחים (Prepared Statements) עם שמות משתנים כמו @Username.
כך, הקלט מהמשתמש עובר ישירות כערך ולא כחלק מקוד SQL, ולכן לא ניתן "להזריק" לתוכו פקודות.
השיטה הזו מאובטחת יותר ומונעת את רוב מתקפות ההזרקה הנפוצות.
שימוש בפרמטרים גם משפר את קריאות הקוד וביצועי מסד הנתונים.
מעבר לכך, מומלץ לא לשמור סיסמאות כטקסט אלא כ־Hash מוצפן.
בהשוואת התחברות – מבצעים Hash על הקלט ומשווים לתוצאה השמורה במסד.