**סיסמאות בקוד:**

אסור בתכלית האיסור לשמור בקוד סיסמאות, טוקנים, api keys, secrets וכהנה וכהנה פיסות מידע רגישות.

הפתרון: שימוש ב-custom-environment-variables.json של מודול ה-node-config שאנו משתמשים בו, כדי למפות משתני סביבה למשתני קונפיג

**logging**

מה קורה כאשר יש שגיאה בשרת? בדרך כלל אנו שולחים שגיאה ללקוח. יש כאן סכנה משום שאנו עלולים לחשוף מידע רגיש להאקרים הנחשפים להודעות השגיאה, מבלי שנתכוון. מצד שני, אנו רוצים לדעת בעצמנו על השגיאות כשהן קורות כדי שנוכל לטפל בהן.

הפתרון: Logging.

כשתקרה שגיאה בשרת, נעשה שני דברים:

1. נשלח ללקוח הודעה לקונית, ולעולם לא הודעה מפורטת על מה הייתה השגיאה.
2. נעשה log להודעת השגיאה המקורית

מקובל להשתמש ב-npm שנקרא winston על מנת ליישם את הלוגינג

רמות חומרב בלוגינג:

* Debug
* Info
* Warning
* Error
* Severe

**CORS**

נרצה להתשמש ב-CORS בסביבת production.

**JWT**

לפני יצירת הטוקן, למחוק את הסיסמא, גם אם היא מוצפנת!

**SALTing passwords**

כדי שכל מיני Rainbow Tables לא ימצאו את הסיסמה המוצפנת שלנו, עלינו "להמליח אותה." כלומר לפזר את מחרוזת ה-salt באופן שווה בין התווים של מחרוזת הסיסמה. הדרך הקלה לעשות זאת ב-Node.js היא

createHmac('md5', salt).update(`${plainTextPassword}`) .digest('hex'); // export has hexa

**SQL Injection**

ראינו בהדגמה איך ניתן ליצור נזקים בקלות לדטהבייס של המערכת אם משרשים מחרוזות כדי לבנות פקודות SQL בצורה דינמית, למשל כך:

const sql = `SELECT \* FROM users WHERE username = '${credentials.username}' AND password = '${credentials.password}'`;

כדי להימנע מנזקים, לא נשרשר מחרוזות לעולם, ובמקום זאת נסמן לדרייבר איפה בשאילתא נרצה לשתול את הפרמטרים, ואילו פרמטרים נרצה לשתול, והוא (הדרייבר) כבר יעשה להם escaping כך שלא נהיה נתונים לסכנה של SQL Injection

**XSS**

Cross site scripting

האקרים עלולים להשתמש בשדות קלט, כדי להכניס קוד HTML זדוני (ספציפית בתוך תגיות script) ואז אם נרנדר את הקלט הזה יותר מאוחר בדפי HTML, הקוד הזדוני שלהם ירוץ בתוך הדפים שלנו.

הפתרון: לעשות striptags לכל הקלט בצורה גורפת, כדאי באמצעות middleware שרץ מיד לאחר express.json express.urlEncoded וכיוב מידלוורים שטוענים נתונים ל-req.body.

**Insecure Direct Object Reference**

הבעיה מופיעה כשמתשמים ב-id שהוא מספרי, למשל שמקורו בשדה auto increment בדטה-בייס.

הפתרון:

* להמנע מלהשתמש ב-id ב-url כשניתן. למשל מסך עריכת פרטי משתמש, פרטי המשתמש נתונים לאפפ ב-jwt ואין שום צורך להציג אותם ב-url
* כמו כן, לא מומלץ בהחלט להשתמש ב - auto increment אלא ב-uuid

**DOS**

Denial Of Service

התקפה על שרת במטרה להעמיס עליו כדי שלא יוכל לתת שירות ללקוחות שלו. כמובן שבתור "בעל העסק" אנו לא רוצים להיות מותקפים בצורה הזו.

הפתרון: שימוש ב-rate limit. אפשר להשתמש ב-middleware שמיישם רייט-לימיט

**DDOS**

Distributed Denial of Service

פתרון: אין באמת פתרון ברמת התוכנה של האפפ. ניתן להשתמש בשירותים חיצוניים למשל Cloudflare